4-10-00

A

JC377 U.S. PTO 04/07/00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Atty. Docket: L&P / 1089A

Applicant: Robert D. Oexman, David B. Scott

Title: CUSTOMIZED MATTRESS EVALUATION SYSTEM

JC520 U.S. PTO 09/544958 04/07/00

CERTIFICATE OF MAILING BY EXPRESS MAIL - 37 CFR 1.10

'Express Mail' mailing label number: EL583224930US
Date of Deposit: April 7, 2000

I certify that this paper or fee (along with the enclosures noted herein) is being deposited with the United States Postal Service 'Express Mail Post Office to Addressee' service under 37 CFR 1.10 on the above date and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

By: Heath R Bandy (person mailing paper)

UTILITY PATENT APPLICATION TRANSMITTAL

BOX PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

This is a request for filing, under 37 CFR § 1.53(b), a(n):

- ☒ **Original (non-provisional) application.**
- ☐ Divisional of prior application Serial No. __, filed on __.
- ☐ Continuation of prior application Serial No. __, filed on __.
- ☐ Continuation-in-part of prior application Serial No. __, filed on __.

PRELIMINARY AMENDMENT/CALCULATION OF FEES

- ☐ Please cancel claims __ without prejudice, and prior to calculating the fees. __ total claim(s), of which __ is(are) independent, is(are) pending after the amendment.
- ☐ Please enter the enclosed preliminary amendment identified below prior to calculating the fees. __ total claim(s), of which __ is(are) independent, is(are) pending after the amendment.
- ☒ **The Fees are Calculated as Follows:**

Fee:	Number of Claims:	In Excess of:	Extra.	At Rate:	Amount:
Total Claims	14	20	0	$18	$0.00
Independent Claims	4	3	1	$78	$78.00
MULTIPLE DEPENDENT CLAIM FEE					
BASIC FEE					$690.00
TOTAL OF ABOVE CALCULATIONS					$768.00
REDUCTION BY 50% FOR FILING BY SMALL ENTITY					
TOTAL					$768.00

ENCLOSURES

☒ **Utility Patent Application Transmittal Form containing Certificate of Mailing By Express Mail Under 37 CFR 1.10.**

☒ **Return Postcard.**

APPLICATION PAPERS

☒ **Utility Patent Application, with: cover sheet, _25_ page(s) specification (including _14_ total claim(s), of which _4_ is(are) independent), and _1_ page(s) abstract.**

☒ **Drawings: _5_ sheet(s) of _informal_ drawings (_6_ total figure(s)).**

☐ Microfiche Computer Program (Appendix).

☐ Nucleotide and/or Amino Acid Sequence, including (all are necessary): Computer Readable Copy, Paper Copy (identical to computer copy), and Statement verifying identity of copies.

☒ **An _Unsigned_ Declaration, Power of Attorney and Petition Form.**

☐ Copy of Executed Declaration, Power of Attorney and Petition Form from prior application identified above.

☐ Certified Copy of priority document(s) identified as attached above.

ADDITIONAL PAPERS

☐ Assignment to __, Recordation Cover Sheet (Form PTO-1595)

☐ Verified Statement to Establish Small Entity Status under 37 CFR 1.9 and 1.27.

☐ Preliminary Amendment (to be entered prior to calculation of fees)

☐ Information Disclosure Statement, __ sheet(s) Form PTO-1449, __ U.S. Patent Reference(s), __ Foreign Patent Reference(s) and __ Other Reference(s)

☒ **Other: Appendix A _6 PAGES_ and Appendix B _52 PAGES_**

CHECKS

☐ A Check of __ for the filing fee.

☐ A Check of __ for the assignment recording fee.

DEPOSIT ACCOUNT AUTHORIZATION

☐ Please charge Deposit Account No. _ in the amount of __.

☐ The Commissioner is authorized to charge any fees under 37 CFR 1.16 and 1.17 which may be required during the entire pendency of the application, or credit any overpayment, to Deposit Account No. __. A duplicate of this transmittal is attached.

☒ **THE PAYMENT OF FEES IS BEING DEFERRED.**

Respectfully Submitted,

Joseph R. Jordan
Reg. No. 25,686

WOOD, HERRON & EVANS, L.L.P.
2700 Carew Tower
Cincinnati, Ohio 45202
(513) 241-2324

JRJ/jlv

EXPRESS MAIL NO.: EL583224930US

APPLICATION
FOR
UNITED STATES PATENT

Applicant: **Robert D. Oexman and David B. Scott**

Title: **CUSTOMIZED MATTRESS SYSTEM**

Wood, Herron & Evans, L.L.P.
2700 Carew Tower
Cincinnati, Ohio 45202
Attorneys
(513) 241-2324

SPECIFICATION

CUSTOMIZED MATTRESS SYSTEM

This application claims priority to U.S. provisional patent application serial no. 60/128,104, filed April 7, 1999, hereby expressly incorporated by reference herein.

Field of the Invention

This invention relates to the designing or selecting of mattresses and mattress and box spring foundation unit assemblies that are most suitable for persons of various body types, and particularly to the providing of customized mattresses or mattress and box spring assemblies for individual persons.

Background of the Invention

Many different mattresses and foundation units are currently available on the market. Each of these may provide acceptable comfort and support for some persons who use them, but all will be less suitable for at least some users. The differences among individual persons with respect to their body frames, physiques and weights and their individual physical characteristics and desires

are large. These differences account for differences in the suitability of particular mattresses or mattress and box spring combinations among members of the public. Presently, there exist systems for determining how pressure may be distributed between the upper supporting surface of a mattress and a person reclining on the mattress. But systems that take into account the support provided by a mattress in maintaining alignment of the spine and other body parts for a particular person or particular type of person are not available.

A need exists for a bedding product providing system that gives users support as well as comfort, particularly a system for correlating the support capabilities of a mattress or mattress system with the support requirements of individual users. There is a particular need for systems for evaluating, designing or selecting particular mattresses or mattress systems for individuals of specific body characteristics and for designing, selecting or customizing such mattresses or mattress systems to meet the support and comfort requirements of such individuals.

Summary of the Invention

An objective of the present invention is to provide specific individuals or persons of specific body types with mattresses or mattress and foundation sets that are particularly suited for optimal comfort and support.

A particular objective of the invention is to provide a system and method which determines both pressure distribution and body support requirements for providing mattress systems to individuals based on the particular body characteristics of the individual.

Further objectives of the present invention are to provide a system and method capable of optimizing pressure characteristics of a person on a mattress and capable of taking into account the support characteristics that the mattress provides for the person.

The present invention attains the above described objectives, overcomes the drawbacks of known systems of the prior art and fills the needs set forth above.

According to the principles of the present invention, data is acquired of the body characteristics of an individual customer. The data preferably includes body dimensions of the person such as, for example, height, torso length, hip width, waist width and shoulder width, as well as the overall weight of the person.

Preferably, the data is collected by providing a pressure sensor array on the surface of a standardized support such as an airbed mattress and collecting pressure data over the two-dimensional array. The overall air mattress pressure is also recorded. In the preferred embodiment, the data is processed by a computer which derives the body characteristics of the individual.

Further, according to the invention, data is provided that relates each of a plurality of bedding products, that is, mattresses and mattress and box spring combinations, having different design properties with body characteristic profiles of persons for whom the bedding product provides correct support. Preferably, this data is collected by causing each of a plurality of representative people to recline on each of the plurality of bedding products, evaluating and quantizing the support provided for that person, and measuring, using a pressure sensor array, the pressure distribution of the person when properly supported. Data

which rates the support provided for such person by the bedding product as well as data which rates the pressure distribution are tabulated for each combination of bedding product and each body profile.

In one embodiment of the invention, body profiles of each of the representative people used to test the bedding products are classified by having each such person recline on a test device that includes a test pad on the air mattress. The test pad and air mattress system is standardized so that the data is taken under conditions that are the same for each of the prospective customers of a bedding product. Air pressure sensors are connected to the air mattress to provide data of the total air pressure in the mattress, or on various zones of the mattress, when a person is lying on the test device. The test pad includes an orthogonal array of pressure transducers, for example 30 x 30 array of pressure sensors, that measure local pressures exerted between the pad and the body of the person lying on the device.

With the preferred embodiment, a large number of body types are tabulated. A large number of bedding products are also evaluated. These bedding products include unique combinations of box spring, inner spring, and various layers of padding or fill, including, for example, bottom fill and top fill. With combinations made up, for example, of four box springs, seven inner springs, four types of bottom fill and two types of top fill, a total of 168 mattress and box spring combinations are possible. Each combination is evaluated with respect to each of several different body types. The product evaluations may be carried out with each of a representative plurality of physically different test persons lying on each product and data being taken to quantitatively rate each

such product with respect to support and pressure distribution it provides to each such test person. Each test person is also tested on the standardized pad and airbed system and the body type of each such test person is automatically classified by the computer. A table evaluating each bedding product for each body type can be generated.

In the preferred embodiment of the invention, rather than providing data for each of a plurality of body types and for each of a plurality of bedding products, 168 bedding products in the example above, certain body characteristics can be correlated to coefficients for each of the various bedding components. For example, one coefficient can be derived from each body type as primarily affecting selection of the box spring, with another coefficient for inner spring, another for bottom fill and another for top fill. It is found that overall weight most affects the box spring selection, while weight distribution most affects inner spring selection. Various body dimensions affect selection of fills or padding.

According to alternative embodiments of the invention, instead of the customer reclining on a pressure array pad on an air mattress, no pressure pad array is used, but the person is asked to provide certain body characteristic information. In one such alternative embodiment, the person is asked to recline on a standardized airbed and an overall pressure reading is taken. The person answers a limited number of questions. The answers to the questic s and the overall pressure reading are correlated by a computer with the mattress that most closely provides the optimal support and minimal pressure to a person most closely matching the body profile. This embodiment is suitable for

providing a customer with the most suitable one of a small number of bedding products, for example four or five products, based on a minimum of questions, such as sex, age, height and whether a person experiences any particular pain. In another such alternative embodiment, no measurements are taken, but only questions are asked of the customer, and the questions are more extensive. Additional questions such as the weight of the person and various clothing sizes of the person are filled out on a questionnaire. This embodiment also is mainly suitable for selecting the best bedding product from among a relatively small number of possible products, for example, four.

The present invention allows a customer to enter a retail mattress store or the like, and purchase a mattress which is customized specifically for that person. The individual will be asked to lie down on an evaluation mattress and/or complete a questionnaire. In the preferred embodiment, where a pressure array pad is used in combination with an air mattress, the pad is positioned atop the standardized evaluation air mattress. The pad includes a plurality of pressure sensors that measure the pressure between the person lying on the mattress and the mattress. The computer scans the pressure data output by the sensors of the pad array and, by scanning the sensors in particular orders and interpreting the sensor outputs, the coordinates of the extremities of the person lying on the mattress are located and body dimensions and other parameters are determined. The pad gathers information from a person lying on the pad, such as the person's height, the person's weight, the distribution of the person's weight or the support pressure value distribution over the various sensors of the array, the area over which the person's weight is distributed, the

width of the person's shoulders, the width of the person's waist area and the width of the person's hips, and the weight born by certain areas of the pad such as that supporting the lumbar area. In particular, data correlated to the lumbar curve of the person's back when in a lying position is derived.

From the measured and derived information, the system's computer preferably generates a plurality of coefficients that are most useful for determining one or more mattress system components that provide the best comfort and support for the customer. A decision as to which mattress system design provides the best comfort and support for a person of a particular body type can be made by any of the various theories supported by professionals. The present invention assumes that the appropriate mattress system for a given body type is that which provides the lowest maximum pressure between the person and the upper support surface of the mattress while providing a spinal curvature that is regarded as proper by knowledgeable professionals.

In the illustrated embodiment of the invention, four coefficients are derived. These include a coefficient that relates to the box spring, one that relates to the innerspring, and two that relate to mattress fill materials, including one that relates to a bottom fill material which includes padding adjacent the innerspring unit, and one that relates to a top fill material which includes additional the quilted ticking and additional padding that provides the outer covering of the mattress. In determining the coefficients, the computer of the system computes the spinal support required for the individual and certain body weights and dimensions from the pressure pad data. The computer calculates and displays a body contact profile, a spinal support profile, body distribution

profile and other comfort and support factors. A graph showing respiratory effort will be generated from the mattress on which the pad is placed. The coefficients that are generated from the evaluation system correlate to components available for manufacture of a mattress or foundation unit that will be custom built according to each customer's unique body shape and weight or to select an available and compatible mattress and foundation unit from stock.

In the preferred embodiment, body type coefficients are produced and correlated with a table of product design parameters to arrive at the optimum bedding product for the user according to the user's particular body type. The correlation table is preferably generated by a method of determining the support characteristics of each test mattress in a mattress retailer's catalog or inventory relative to the weight and size profiles of a plurality of individual test persons. The method may, for example, include the steps of (a) measuring the weight distribution profiles of numerous different height, weight and shaped test persons; (b) measuring and determining the optional deflection profile of those same persons for optimal support; (c) inputting the information of steps (a) and (b) into a computerized control; (d) locating a selected mattress in a test apparatus; (e) applying a weight profile load of selected test profile persons to pistons connected to independently movable pressure plates of the test apparatus, which pressure plates are positioned and sized on the mattress so as to mimic the shape of the selected persons; (f) measuring the deflection of each pressure plate into the test mattress; and (g) comparing the measured deflection characteristics of the test mattress to the optional support deflections for the test profile persons. Alternatively, the table may be generated by a

method of determining the support characteristics of each test mattress relative to human weight and size test person profiles by (a) measuring the weight distribution profile of numerous different height, weight and shaped test persons; (b) measuring and determining the optional deflection profile of those same persons for optimal support; (c) inputting the information of (a) and (b) into a computerized control; (d) locating a selected mattress in a test apparatus; (e) applying optimum deflection characteristics profile of a selected test profile person to that mattress by applying pressure to pistons connected to independently movable pressure plates of test apparatus, which pressure plates are positioned and sized so as to mimic the shape of the selected test profile person; and (f) measuring the pressure on the pistons to obtain this optimum deflection.

Brief Description of the Drawings

The objectives and features of the invention will become more readily apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

Fig. 1 is a diagrammatic representation of the system according to a presently preferred embodiment of this invention; and

Fig. 2 is a representative display of pressure profile information generated and utilized in this invention.

Fig. 3 is a perspective view of a mattress product rating subsystem for use with the system of **Fig. 1**.

Fig. 4 is a cross-sectional view of the subsystem of **Fig. 3**.

Figs. 5 and **6** are diagrams of the operation logic of the mattress product rating subsystem of **Figs. 3** and **4**.

Detailed Description of the Invention

Fig. 1 schematically shows a customized mattress providing system 10 according to one embodiment of the present invention. The system 10 is a customized mattress evaluation, selection or designing system and includes an automated body type determination subsystem 25 by which data characteristic of a person's body shape or body type is generated. The system 10 also includes a bedding product rating data source or subsystem 26 and a mattress selection or designing subsystem 27. The mattress selection or designing subsystem 27 includes a manual or automated system by which mattress product design information from the bedding product rating data source 26 is correlated with specific body type information from the system 25 for an individual bedding customer to arrive at a design for a mattress produce that has been custom designed or selected for that customer.

The body type determination subsystem 25 includes a support surface 12 which may be typically located in a retail mattress store. The support surface 12 is preferably an air bed or some other structure that can be assured of producing standardized support characteristics that are consistent over time. Such an air bed may typically have a plurality of sealed zones, typically four in number, with a specified and constant amount of pressure in each. Pressure of the air bed zones is typically at or near twelve inches of water pressure in an unloaded state. The subsystem 25 further includes a pressure sensor array pad 14, which is positioned on the top of the support surface 12. The pad 14 includes an array

of pressure sensors 30, of for example 900 or 1,024 in number. Preferably the array is rectangular with the sensors 30 arranged in a plurality of orthogonal columns and rows. The sensors 30 are calibrated to each produce a zero output when no person is reclining on the pad, but to each produce an output signal proportional to the pressure exerted on an area of the pad at which the sensor is located

In use, a person 16, for example an individual bedding customer, lies on the pad 14 while it is positioned on top of the support surface 12. Pressure on each of the individual pressure sensors 30 of the pad 14 is measured by the sensors 30 and a pressure signal containing pressure profile array data 18 is generated. Optionally, pressure sensors (not shown) in the zones of the airbed support 12 may also generate signals of support pressure data 31 of each of the airbed zones. The signals 18 and 31 are relayed to a computer 32 or other processor which records digital information of the pressure profile 18, and pressure data 31 if used, of the person 16 lying on the support surface 12.

The computer 32 derives information from the digital information records by processing the data. The information which can be derived form the pressure data includes, for example, that of the person's height, weight, distribution of weight, width of shoulders, width of waist area, width of hips and the lumbar curve of the person's back when in the lying position. The pad 14, along with the sensors 30 and the software or algorithm for generating the pressure profile is commercially available from Vista Medical of 120 Maryland Street, Winnipeg Manitoba, Canada. The Vista Medical Force Sensing Array: Pressure Mapping System is described by its manufacturer as a clinical tool used to assess

pressure distribution and positioning. The information from the force sensing array pad or mat is displayed on a computer screen as easy to understand color graphics and data, a sample 60 of which is shown as **Fig. 2**, with different colors codes 61 used to illustrate different pressure readings. Numerous display options of the system may include a table 62 of numerical pressure values and statistical data, and three-dimensional wire-grid representation 63 of the pressure distributed over the array of the pad 14.

The pressure profile data generated for the individual person is then used, in a preferred embodiment, to generate specific body shape coefficients or other body type parameters 33 which define mattress design parameters. In a presently preferred embodiment, the mattress design parameters include a box spring coefficient, an innerspring coefficient, a bottom fill coefficient and an upper fill coefficient. The program in the computer 32 which generates these mattress design parameters in a presently preferred embodiment is written the Microsoft VISUAL BASIC computer language, a source code of which is set forth in Appendix A to this application.

In particular, the program in computer 32 reads in the data from the sensors 30 by scanning the outputs of each of the sensors of the rows and columns of the array of the pad 14 with a user reclining on the pad. In the program set forth in Appendix A, note that the rows are defined as being parallel to the body of the user 16 while the columns are defined as transverse to the body of the user 16. First, the output of each of the sensors 30 of the array is tested against a minimum threshold to determine the area of the pad that is supporting the weight of the user. This area is represented by the number of

sensors having a reading above the threshold. Second, the pressure outputs of each of the sensors 30 are summed and from this sum the weight of the user is calculated. Third, the weight of the user is divided by the number of sensors supporting the user to arrive at an average pressure per unit area or weight distribution.

Next, the outputs of the sensors are scanned from the ends of the rows to locate the extreme head and foot positions of the user on the pad. From these head and foot positions, the height of the user is calculated. The crotch height of the user is also calculated from the head and foot positions, using statistically averages body proportions. Then, from the head and foot positions, a calculation is made of the user's shoulder height, using statistically average body proportions, and at this height the shoulder positions of the user are determined by scanning sensor outputs from opposite ends of the columns at this shoulder height, and from the shoulder positions the user's shoulder width is calculated. In addition, positions of the shoulders, such as the top of the shoulders, the center of the shoulders and the center of the hips can be determined by scanning the sensor outputs and computing intermediate positions. Such additional determinations are optional, and may be used to determine the position of the user on the pad and to make corrections in the event that the user is not reclining on the center of the pad.

Further, lumber support is determined. This determination uses the crotch height and waist center calculations to define a rectangular area of the pad that is supporting the lumbar portion of the users back. The total pressure readings of the sensors 30 over this lumbar area is calculated and this total

lumbar force is divided by the average pressure calculation of the entire user support area to produce a lumbar curve coefficient. This area provides information that can be correlated with empirical data from which correlation the lumbar curve of the user can be determined.

From the above measurements and calculations, four coefficients are derived which define the body type parameters that serve as the mattress design parameters discussed above. One is a box spring coefficient which is directly proportional to the total calculated weight of the user. Another is the innerspring coefficient which is proportional to the total weight of the user divided by the average pressure on the sensors 30 in the support are of the pad 14, which is proportional to the area of support or total number of sensors measuring pressure. Additionally, a bottom fill coefficient is calculated by dividing the total weight of the user by the user's shoulder width and a top fill coefficient is set as directly proportional to the lumbar curve coefficient. All of the coefficients are multiplied by constants.

The specific mattress design parameters or above-identified co-efficients are used to specially make a mattress or mattress and box spring combination optimally compatible with the individual customer. Alternatively, based upon the mattress design parameters or co-efficients, a specific mattress available from stock which closely approximates or satisfies the mattress design parameters or co-efficients is selected for the individual. The designing or selection of a mattress may be carried out manually or automatically by correlating the body type or shape parameter data 33 generated by the subsystem 25 from the individual customer with product vs. body type correlation data 34.

The body type correlation data 34 is preferably produced by the manufacturer of the system 10 with bedding product rating data source or subsystem 26 by evaluating various bedding products of a bedding manufacturer and producing a table that correlates various body types, defined by unique combinations of body type coefficients, with support characteristics of the different bedding products. Preferably, the correlation data is generated by generating body type parameter data 48 from a plurality of test persons using a subsystem 25a that is identical to the body type determination system 25 described above. The same test persons or a simulation thereof are then caused to subject the same supporting pressure distribution or support deflection to bedding products having different combinations of the components of box spring, inner spring, top fill and bottom fill. The suitability of each product for supporting the loads of each test person are then evaluated by a professional evaluation system 49, which can be under the guidance of medical professionals. A computer 50 may then be used to correlate the data 48 of the body types of each test person with the rating data 49 of the various bedding products. The mattress design or selection subsystem 35 matches the body type parameter data 35 from the individual customer with the bedding product vs. body type correlation data 34 to arrive at a customized mattress design 22.

A system for assessing the specific deflection characteristics of a mattress can be utilized to determine which specific mattress design is appropriate to satisfy the mattress design parameters. Such a mattress design system is envisioned to include a machine 70, as illustrated in **Figs. 3** and **4**, which has a plurality of spaced platens 71 arranged over the upper surface of

selected mattress product 72. The machine 70 can be controlled to simulate each of the test persons who 46 so that the deflection characteristics of each mattress product with respect to each of the platens 71 can be measured as the platens are depressed downwardly into the mattress to determine the force versus deflection data for the particular mattress. Optimally, the multiple platens are arranged in the shape of a human body with forces in proportion to the human body and physique of each of the test persons. The deflection is controlled to be optimal from a support or optimal lumbar curve standpoint. In this way, each bedding product is rated for suitability for the support of persons having each of the body types defined by the body type parameters that the test persons produced. For each test person and bedding product, the ratings of the various products seeks to minimize support pressure, as measured by a pad 14 between the mattress 72 and the platens 71, while optimizing the lumbar curve of the test person on the product. A VISUAL BASIC computer program for controlling the machine 70 is set forth in Appendix B to this application.

The machine 70 is a piece of testing equipment designed to optimize the performance of mattresses or other bedding products for particular body types or styles defined by use of the subsystem 25. This machine 70 can measure either mattress deflection based on a predetermined pressure load or pressure based on a predetermined mattress deflection. Each of the platens 71 is driven by a pressure piston so as to mimic the shape of a portion of the human body. The pressure plates or platens 71 are strategically placed and shaped so replicate the shape of a human body when placed on a mattress or other bedding product, and are driven by piston pressure to replicate the body type

characteristics of various test persons. The bedding product is rated for pressure distribution and support by placing it on the testing table and activating the pressure pistons to move the pressure plates downwardly into contact a pressure array pad 14 on the top surface of the bedding product being rated. Computer 48 provides input to the pressure pistons so that the pressure plates apply pressure to the bedding product in a desired pattern so as to mimic a specific body type lying on the bedding product. These inputs are varied to rate the reactions of different types of bedding products with different pressure patterns representative of different body types. The deflection of the individual springs reacting to a predetermined load or configuration of load are used to rate mattresses which have a desired set of deflection characteristics when a predetermined pressure pattern is placed on the bedding product. The ratings particularly take into account lumbar support or production of the optimal lumbar curve as well as a minimization of the maximum support pressure between the user and the mattress.

Additionally, the testing apparatus may determine which type of individual is best suited for a particular construction of bedding product. In this testing mode, the pressure pistons are moved a fixed distance into the bedding product corresponding to the distance a particular type of person would move the springs of the bedding product. A computer then reads the pressure on each of the pressure pistons. This information can be used to manufacture bedding products which have desired deflection characteristics at certain places on the bedding product.

From the above disclosure of the general principles of the present invention and the preceding detailed description of at least one preferred embodiment, those skilled in the art will readily comprehend the various modifications to which this invention is susceptible. Therefore, we desire to be limited only by the scope of the following claims and equivalents thereof.

We claim:

1. A method of providing a custom bedding product for an individual person comprising the steps of:

inputing into a digital processor body type data containing weight and size information of the individual person;

with the processor, evaluating body type data and calculating body type parameters for the individual person;

correlating the body type parameters with bedding product evaluation data and determining bedding product parameters specifying the design of a bedding product for providing suitable pressure distribution and lumbar support for the individual person.

2. The method of claim **1** wherein the inputing of the body type data into the processor includes:

providing a stable support surface;

providing a pressure sensor array on the top of the support surface, the array including a plurality of pressure sensors each adapted to measure pressure exerted against an area of the pad by a person reclining on the pad; and

with the individual person reclining on the pad, generating the body type data characteristic of the individual person.

3. The method of claim **2** wherein:

the stable support surface includes an airbed having at least one zone inflated to a standardized pressure.

4. The method of claim **2** wherein:

the bedding product parameters include at least one box spring coefficient indicative of a suitable box spring for the individual person, at least one innerspring coefficient indicative of a suitable inner spring for the individual person, and at least two pad coefficients indicative of at least two suitable pad layers for the individual person.

5. The method of claim **2** wherein:

the bedding product parameters include at least one coefficient responsive to the total weight of the individual person and the bedding product providing step includes the step of determining a box spring component of the bedding product based on the at least one coefficient.

6. The method of claim **2** wherein:

the bedding product parameters include at least one coefficient responsive to the weight distribution area of support of the individual person and the bedding product providing step includes the step of determining an innerspring component of the bedding product based on the at least one coefficient.

7. The method of claim **2** wherein:

the body type parameters include at least one coefficient responsive to a selected body width of the individual person and the bedding product providing step includes the step of determining a mattress fill component of the bedding product based on the at least one coefficient.

8. The method of claim **2** wherein:

the body type parameters include at least one coefficient responsive to the lumbar support of the individual person and the bedding product providing step includes the step of determining a box spring component of the bedding product based on the at least one coefficient.

9. The method of claim **1** wherein the inputing of the body type data into the processor includes:

providing an airbed having at least one zone inflated to an initial standardized pressure;

with the individual person reclining on the airbed, measuring pressure of air in the airbed and generating therefrom at least some of the body type data characteristic of the individual person.

10. The method of claim **9** wherein the inputing of the body type data into the processor further includes:

obtaining body type information from information provided by the individual person and entering the information into the computer.

11. The method of claim **1** wherein the inputing of the body type data into the processor includes:

obtaining body type information from information provided by the individual person and entering the information into the computer.

12. A system for customization design of a bedding product for an individual person, the system comprising:

a support surface which provides a constant pressure of support;

a pad positioned atop the support surface;

a plurality of pressure sensors located on the pad, the pad and support surface adapted for the individual person to lie down thereon;

a processor electrically coupled to the pressure sensors to record the pressure data detected by the respective pressure sensors;

the processor being programmed with an evaluation algorithm to process the pressure data and produce a pressure profile for the individual person on the pad and in response to the pressure profile to calculate specific body shape parameters correlated to components of a bedding product providing suitable pressure distribution and lumbar support, the parameters including at least one box spring coefficient indicative of a suitable box spring for the individual person, at least one innerspring coefficient indicative of a suitable inner spring for the individual person, and at least two pad coefficients indicative of at least two suitable pad layers for the individual person.

13. The method of determining the support characteristics of a test mattress relative to a human weight and size profile, which method comprises:

(a) measuring the weight distribution profiles of numerous different height, weight and shaped persons;

(b) measuring and determining the optional deflection profile of those same persons for optimal support;

(c) inputting the information of steps (a) and (b) into a computerized control;

(d) locating a selected mattress in a test apparatus;

(e) applying a weight profile load of a selected test profile person to pistons connected to independently movable pressure plates of the test apparatus, which pressure plates are positioned and sized on the mattress so as to mimic the shape of the selected person;

(f) measuring the deflection of each pressure plate into the test mattress; and

(g) comparing the measured deflection characteristics of the test mattress to the optional support deflections for the test profile person.

14. The method of determining the support characteristics of a test mattress relative to a human weight and size profile, which method comprises:

(a) measuring the weight distribution profile of numerous different height, weight and shaped persons;

(b) measuring and determining the optional deflection profile of those same persons for optimal support;

(c) inputting the information of (a) and (b) into a computerized control;

(d) locating a selected mattress in a test apparatus;

(e) applying optimum deflection characteristics profile of a selected test profile person to that mattress by applying pressure to pistons connected to independently movable pressure plates of test apparatus, which pressure plates are positioned and sized so as to mimic the shape of the selected test profile person; and

(f) measuring the pressure on the pistons to obtain this optimum deflection.

CUSTOMIZED MATTRESS SYSTEM

Abstract of the Disclosure

A customized mattress evaluation system allows for uniquely designed mattresses based upon a particular customer's physical attributes. The system allows a retail mattress store to collect data from a sensor pad positioned on top of a support surface to generate a pressure profile for that person. The pressure profile and other information are used to generate specific mattress design parameters or co-efficients which are then utilized in designing a specific mattress uniquely customized for that person. Body type coefficients characteristic of an individual customer are correlated with coefficients developed for test persons for which various bedding products have been optimized. The optimization includes the rating of various bedding products for various body types by minimizing support pressures across the mattress and optimizing lumbar support for desired spinal curvature.

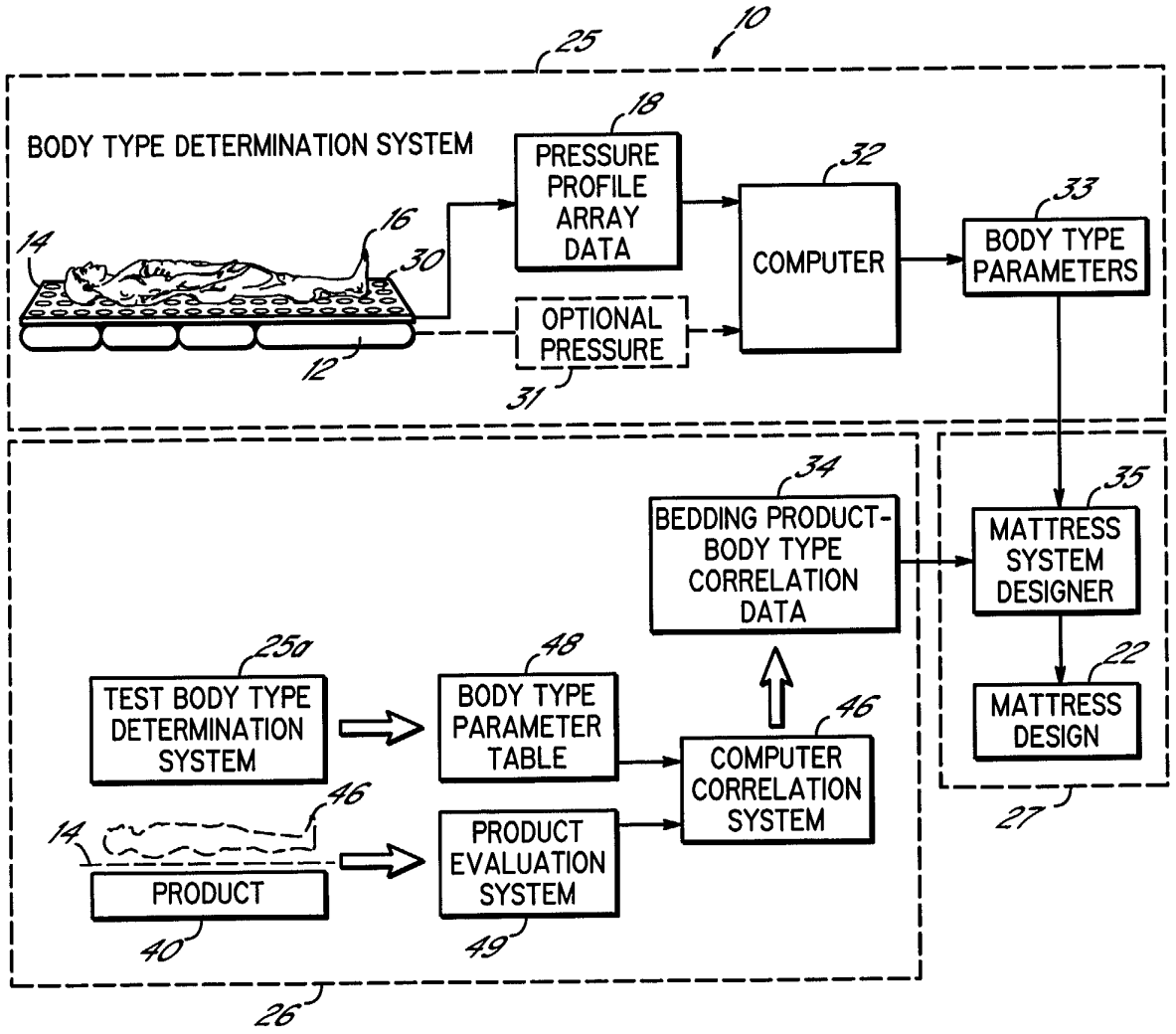


FIG. 1

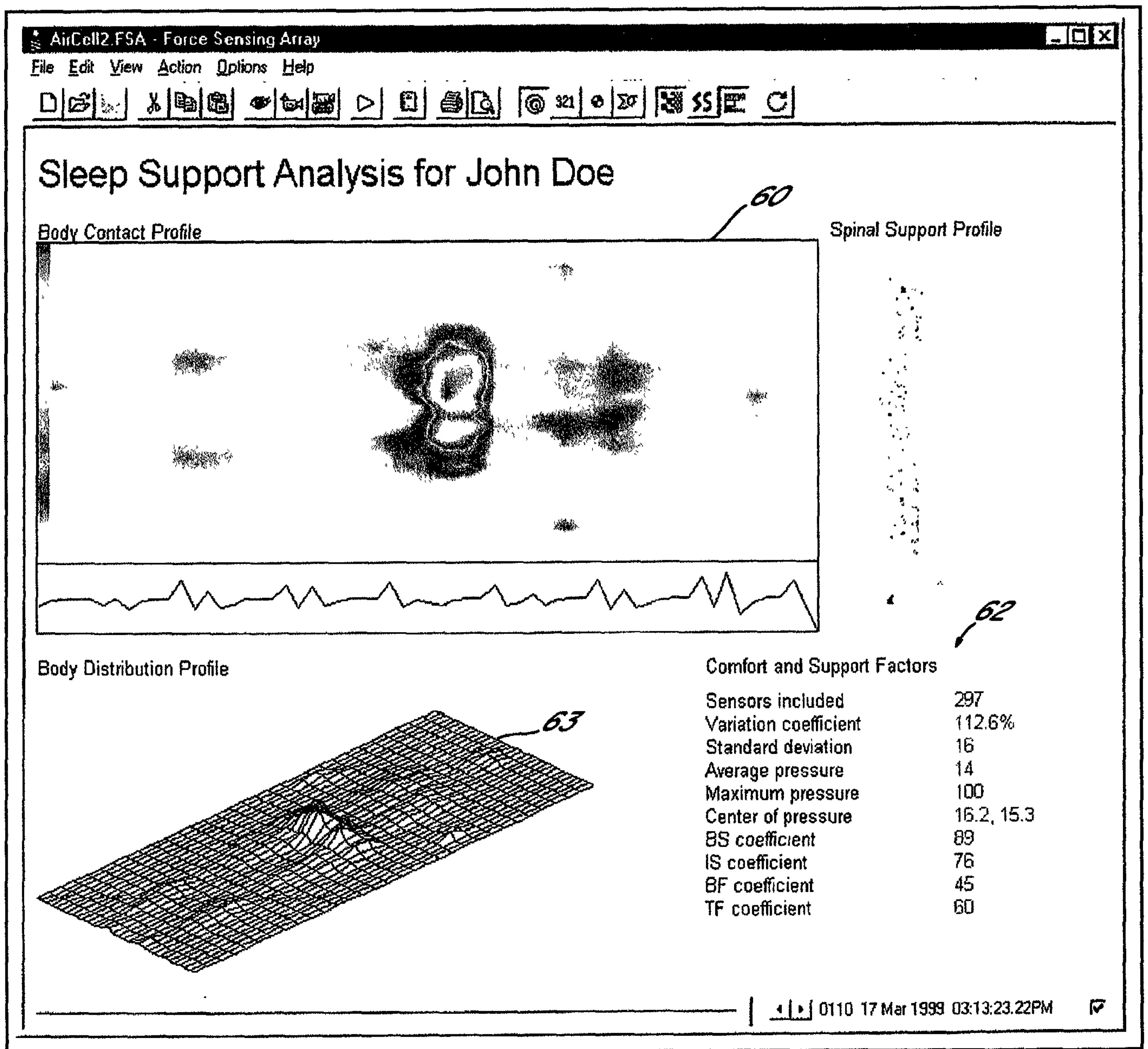

AirCell2.FSA - Force Sensing Array
File Edit View Action Options Help
Sleep Support Analysis for John Doe
60
Body Contact Profile
Spinal Support Profile
62
Body Distribution Profile
63
Comfort and Support Factors
Sensors included 297
Variation coefficient 112.6%
Standard deviation 16
Average pressure 14
Maximum pressure 100
Center of pressure 16.2, 15.3
BS coefficient 89
IS coefficient 76
BF coefficient 45
TF coefficient 60
0110 17 Mar 1999 03:13:23.22PM

FIG. 2

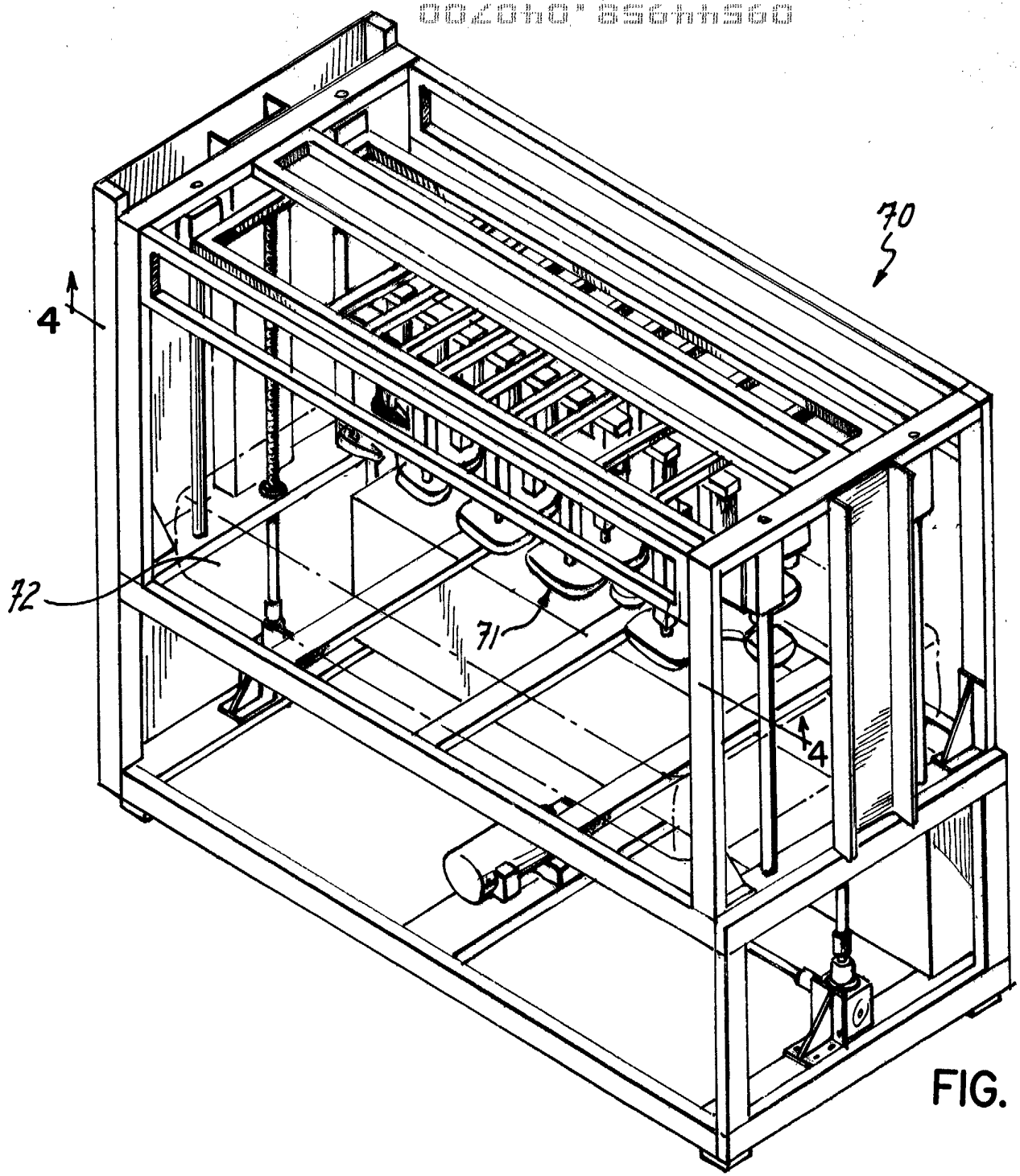


FIG. 3

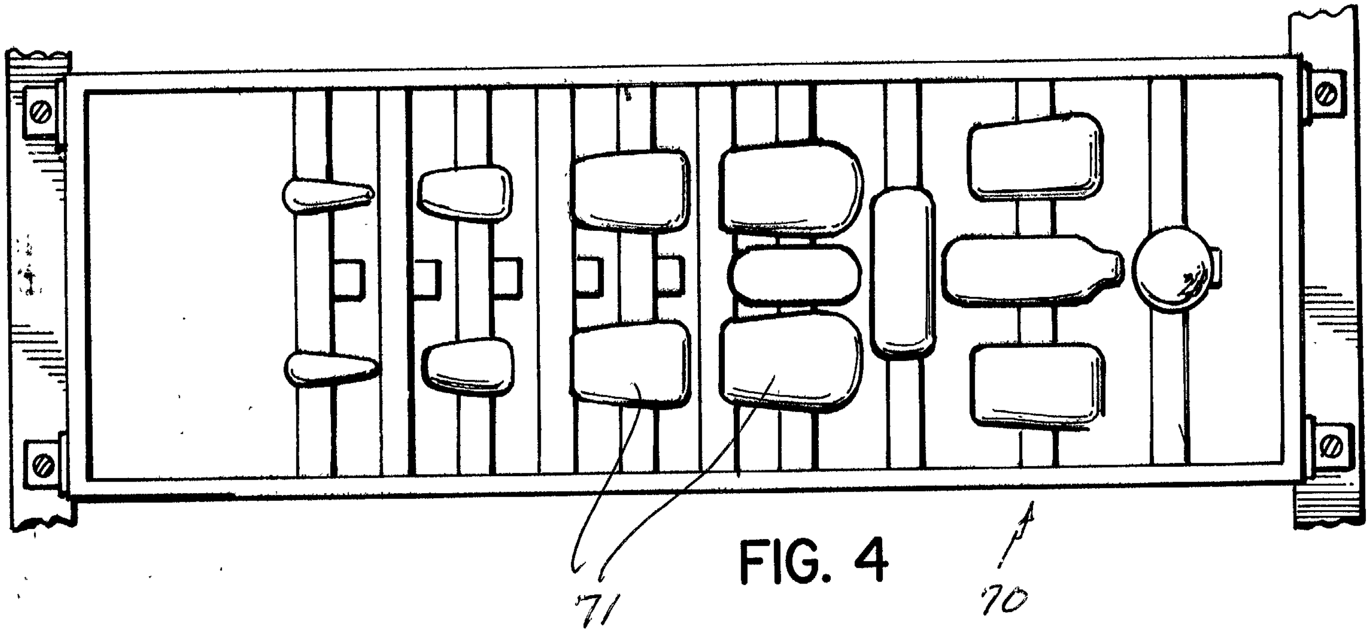


FIG. 4

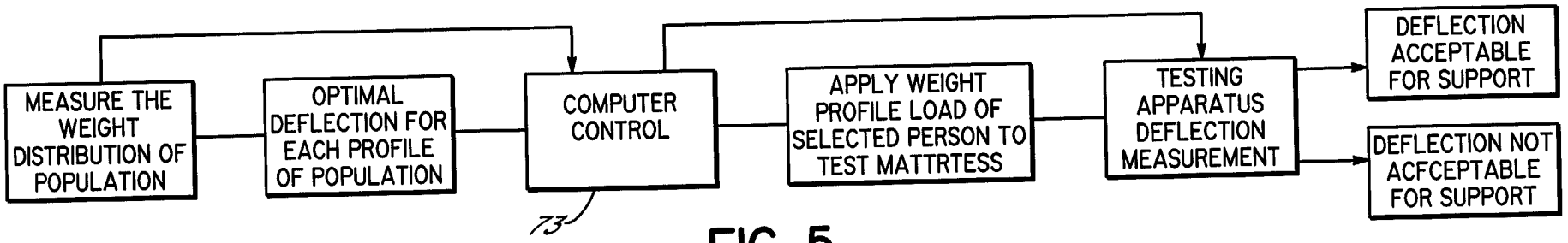


FIG. 5

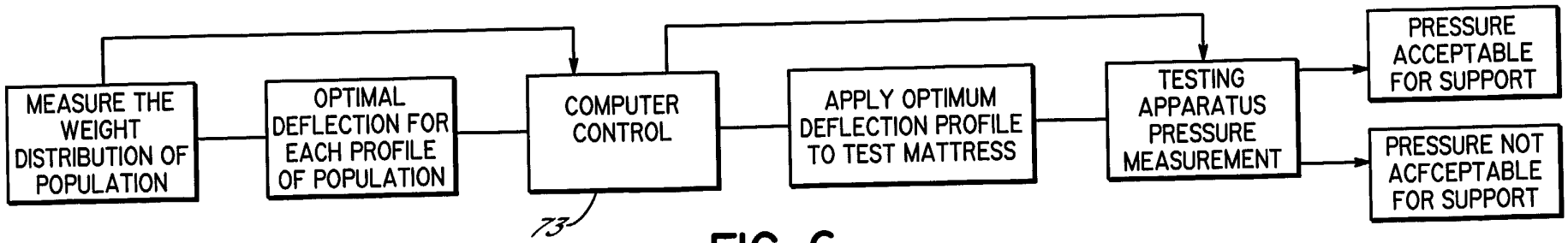


FIG. 6

DECLARATION, POWER OF ATTORNEY, AND PETITION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

CUSTOMIZED MATTRESS EVALUATION SYSTEM

the specification of which (check one below):

(✔) is attached hereto.

( ) was filed on ___ as Application Serial No. ___ or Express Mail No. ___, and was amended on ___ (if applicable).

( ) was filed on ___ as PCT International Application No. ___ , and as amended under PCT Article 19 on ___ (if any).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s) Priority Claimed?

(Number)	(Country)	Day/Month/Year Filed	( ) Yes ( ) No
(Number)	(Country)	Day/Month/Year Filed	( ) Yes ( ) No
(Number)	(Country)	Day/Month/Year Filed	( ) Yes ( ) No

I hereby claim the benefit under Title 35, United States Code, §120 and/or §119(e) of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations §1.56, which became available between the filing date of the prior application and the national or PCT international filing date of this application.

(Serial No.)	(Filing Date)	(Status: Patented, Pending, or Abandoned)
60/128,104	April 7, 1999	Pending

I hereby appoint John D. Poffenberger (R. No. 20,245), Bruce Tittel (R. No. 22,324), Donald F. Frei (R. No. 21,190), David J. Josephic (R. No. 22,849), A. Ralph Navaro, Jr. (R. No. 23,050), David S. Stallard (R. No. 25,930), J. Robert Chambers (R. No. 25,448), Gregory J. Lunn (R. No. 29,945), Kurt L. Grossman (R. No. 29,799), Clement H. Luken, Jr. (R. No. 32,742), Thomas J. Burger (R. No. 32,662), Gregory F. Ahrens (R. No. 32,957), Wayne L. Jacobs (R. No. 35,553), Kurt A. Summe (R. No. 36,023), Kevin G. Rooney (R. No. 36,330), Keith R. Haupt (R. No. 37,638), Theodore R. Remaklus (R. No. 38,754), Thomas W. Humphrey (R. No. 34,353), Joseph R. Jordan (R. No. 25,686), C. Richard Eby (R. No. 25,854), David E. Pritchard (R. No. 38,273), David H. Brinkman (R. No. 40,532), J. Dwight Poffenberger, Jr. (R. No. 35,324), Beverly A. Lyman (R. No. 41,961), A. Ralph Navaro III (R. No. 46,207), Scott A. Stinebruner (R. No. 38,323), Kristi L. Davidson (R. No. 44,643), P. Andrew Blatt (R. No. 44,540), David E. Franklin (R. No. 39,194), Herbert C. Brinkman (R. No. 16,955), all of Wood, Herron & Evans, L.L.P., 2700

Carew Tower, 441 Vine Street, Cincinnati, OH 45202-2917, telephone no. (513) 241-2324, my attorneys, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith. Address all correspondence and telephone calls to

Joseph R. Jordan
Wood, Herron & Evans, L.L.P.
2700 Carew Tower
441 Vine Street
Cincinnati, OH 45202-2917
Telephone (513) 241-2324

Wherefore I pray that Letters Patent be granted to me for the invention or discovery described and claimed in the foregoing specification and claims, and I hereby subscribe my name to the foregoing specification and claims, declaration, power of attorney, and this petition.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of Inventor Robert D. Oexman

Inventor's Signature _____ Date _____

Residence City/State Carthage, Missouri 64836 Citizenship USA

Post Office Address 1141 South Sunset Lane

Full name of Inventor David B. Scott

Inventor's Signature _____ Date _____

Residence City/State Carthage, Missouri 64836 Citizenship USA

Post Office Address 1582 South Plum Lane

APPENDIX A

```
'Oex2000 - Dbs01
'Revision 2.0.0
'Body Detection and Measurement Algorithms

'Written by David B. Scott

Option Explicit
Dim fsa_cb  As Integer
Dim fsa_is As Integer
Dim fsa_bf As Integer
Dim fsa_tf As Integer
Dim fsa_ss As Integer
Dim PSW As Double
Dim PHW As Double
Dim Barray(0 To 2000) As Long
Dim NewData As Variant
'Storage for easier reference later
Dim NumRows As Long
Dim NumColumns As Long
Dim Stopit As Integer
Dim Head As Integer
Dim Feet As Integer
Dim FSASum As Double
Dim FSAAverage As Double
Dim FSASensors As Long
Dim ShoulderWidth As Double
Dim FSAWeight As Double
Dim FSAHeight As Double
Dim FSAIspring As Double
Dim datacall As Long
Const MARRAY As Integer = 10
Private Type Coefs
    'coefficients
    BFcoefa(1 To MARRAY) As Double
    CBcoefa(1 To MARRAY) As Double
    TFcoefa(1 To MARRAY) As Double
    IScoefa(1 To MARRAY) As Double
    SSprofa(1 To MARRAY) As Double
End Type
Dim carray As Ccefs
Dim cindex As Integer
'Array transfer from Pad Data
Public Function Put_FSAData(ByVal element As Long, ByVal index As Long) As Double
    Dim i As Integer
    Dim x As Double
    On Error Resume Next
    Put_FSAData = -1
    If index < 0 Then Exit Function
    If index > 2000 Then Exit Function
    NewData = True
    x = 0
    Barray(index) = element
    If element Then datacall = datacall + 1
    For i = 0 To index
        x = x + Barray(i)
    Next i
    Put_FSAData = x
    If index = 1023 Then
        Crunchit
    End If
End Function
'send cb value when requested
Public Property Get CBcoef() As Variant
Dim atemp As Double
Dim j As Integer
On Error Resume Next
If NewData Then
    Call Crunchit
```

```
    NewData = False
End If
atemp = 0
For j = 1 To MARRAY
    atemp = atemp + carray.CBcoefa(j)
Next j
CBcoef = Format$(atemp / MARRAY, "0")
End Property
'send is val when requested
Public Property Get IScoef() As Variant
Dim atemp As Double
Dim j As Integer
On Error Resume Next
If NewData Then
    Call Crunchit
    NewData = False
End If
atemp = 0
For j = 1 To MARRAY
    atemp = atemp + carray.IScoefa(j)
Next j
IScoef = Format$(atemp / MARRAY, "0")
End Property
'send bf value when requested
Public Property Get BFcoef() As Variant
Dim atemp As Double
Dim j As Integer
On Error Resume Next
If NewData Then
    Call Crunchit
    NewData = False
End If
atemp = 0
For j = 1 To MARRAY
    atemp = atemp + carray.BFcoefa(j)
Next j
BFcoef = Format$(atemp / MARRAY, "0")
End Property
'send tf value when requested
Public Property Get TFcoef() As Variant
Dim atemp As Double
Dim j As Integer
On Error Resume Next
If NewData Then
    Call Crunchit
    NewData = False
End If
atemp = 0
For j = 1 To MARRAY
    atemp = atemp + carray.TFcoefa(j)
Next j
TFcoef = Format$(atemp / MARRAY, "0")
End Property
'send spine position when requested
Public Property Get SSprof() As Variant
Dim atemp As Double
Dim j As Integer
On Error Resume Next
If NewData Then
    Call Crunchit
    NewData = False
End If
atemp = 0
For j = 1 To MARRAY
    atemp = atemp + carray.SSprofa(j)
Next j
SSprof = Format$(atemp / MARRAY, "0")
End Property
```

```
'Actual Mathmatics for Body Detection and Measurements
Private Sub Crunchit()
    Dim r As Integer
    Dim c As Integer
    Dim InARow As Integer
    Dim DTemp As Double
    Dim fsa As Integer
    Dim darray(32, 32) As Double
    Dim Lumbar As Double
    Dim CrotchHeight As Integer
    Dim CData(1 To 32, 1 To 32)
    Dim UnitMultiplier As Double
    Dim Filter As Double
    Dim Center As Integer
    Dim Cfirst As Integer
    Dim TorsoCenter As Integer
    Dim WaistCenterSum As Integer
    Dim WaistAverage As Double
    Dim Filter2
    On Error Resume Next

    UnitMultiplier = 0.392156862745098

    fsa = 0
    FSASum = 0
    FSASensors = 0

    NumColumns = 32
    NumRows = 32

    Filter = 0.75
    Filter2 = 2
    'Decode one-dimensional data into three-dimensional form
    For c = 1 To NumColumns
        For r = 1 To NumRows

            DTemp = Barray(fsa) * UnitMultiplier
            CData(c, r) = DTemp
                 If DTemp > Filter Then
                    FSASum = FSASum + DTemp
                    If DTemp > Filter2 Then FSASensors = FSASensors + 1
                End If
             fsa = fsa + 1
          Next r
    Next c
    If FSASensors = 0 Then GoTo no_body
    'Compute Sensor Average Pressure
    FSAAverage = FSASum / FSASensors
    'Compute theoretical weight
    FSAWeight = FSASum * 0.0155
    ' if the person is less than 40 lbs - abort
    If FSAWeight < 40 Then GoTo no_body
    'set up first cb factor..
    fsa_cb = FSAWeight * 3.5
    'Find the Head or top active sensor
    Stopit = False
    For c = 1 To NumColumns
        For r = 1 To NumRows
            If CData(c, r) > Filter2 Then Stopit = True
            If Stopit Then Exit For
        Next r
        If Stopit Then Exit For
    Next c
    Head = c
    'Find the feet or bottom active sensor
    Stopit = False
    For c = NumColumns To 1 Step -1
        For r = NumRows To 1 Step -1
```

```
            If CData(c, r) > Filter2 Then Stopit = True
            If Stopit Then Exit For
        Next r
        If Stopit Then Exit For
    Next c
    Feet = c
    'Calculate estimated Height based on Head & Feet detection
    FSAHeight = (2 + (Feet - Head)) * 2.25
    If FSAAverage = 0 Then GoTo error_out
'select crotch height/shoulder width based on calculated height
'from statistical ave values
Select Case (2 + (Feet - Head))      'note #of sensor rows not inches
    Case 25                          -
        CrotchHeight = 11
        ShoulderWidth = 19
    Case 26
        CrotchHeight = 12
        ShoulderWidth = 20
    Case 27
        CrotchHeight = 12
        ShoulderWidth = 21
    Case 28
        CrotchHeight = 13
        ShoulderWidth = 22
    Case 29
        CrotchHeight = 13
        ShoulderWidth = 24
    Case 30
        CrotchHeight = 14
        ShoulderWidth = 24
    Case 31
        CrotchHeight = 14
        ShoulderWidth = 25
    Case 32
        CrotchHeight = 15
        ShoulderWidth = 26
    Case 33
        CrotchHeight = 15
        ShoulderWidth = 27
    Case 34
        CrotchHeight = 15
        ShoulderWidth = 27
    Case 34
        CrotchHeight = 15
        ShoulderWidth = 27
    Case Else
        CrotchHeight = 31
        ShoulderWidth = 31
End Select
    'inter spring value set based on fsa ave weight
    FSAIspring = FSAWeight / FSAAverage
    fsa_is = FSAIspring * 100
    'look for top of shoulders (not used at this time)
    Stopit = 0
    InARow = 0
    For c = 1 To 16
        If c > 32 Then GoTo error_out
        If c < 1 Then GoTo error_out
        For r = 1 To NumRows
            If CData(c, r) > Filter Then
                InARow = InARow + 1
                If InARow < Stopit Then Stopit = InARow
            Else: InARow = 0
            End If
        Next r
        If Stopit < 12 Then Exit For        '12 in a row is down past the head
        Stopit = 0
    Next c
```

```
    'look for center of shoulders (not used at this time)
    If c < 1 Then c = 1
    Center = 0
    Cfirst = 0
    Stopit = False
    For r = 1 To NumRows
        If CData(c, r) > Filter Then
            Center = Center + 1
            If Cfirst = 0 Then Cfirst = r
        End If
    Next r
    fsa_bf = (((FSAWeight / ShoulderWidth)) * 45)
    Center = 0
    Cfirst = 0
    Stopit = False
    'find center of hips
    c = CrotchHeight + 1
    For r = 1 To NumRows
        If CData(c, r) > Filter Then
            Center = Center + 1
            If Cfirst = 0 Then Cfirst = r
        End If
    Next r
    TorsoCenter = Cfirst + (Center / 2)
    WaistCenterSum = 0
    'look at the lumbar area for weight
    For c = CrotchHeight - 2 To CrotchHeight
        For r = TorsoCenter - 5 To TorsoCenter + 5
            If c - 4 < 1 Then GoTo error_out
                WaistCenterSum = WaistCenterSum + CData(c - 4, r)
        Next r
    Next c
    'find lumbar
    WaistAverage = (WaistCenterSum / 33) / FSAAverage
    Lumbar = WaistAverage
    fsa_tf = Lumbar * 100
    fsa_ss = 0
    If Lumbar > 1 Then fsa_ss = 1
    If Lumbar > 1.1 Then fsa_ss = 2
    If Lumbar > 1.2 Then fsa_ss = 3
    If Lumbar > 1.3 Then fsa_ss = 4
    If Lumbar > 1.4 Then fsa_ss = 5
    If Lumbar > 1.5 Then fsa_ss = 6
    If Lumbar > 1.6 Then fsa_ss = 7
    If Lumbar > 1.7 Then fsa_ss = 8
    If Lumbar > 1.8 Then fsa_ss = 9
    GoTo end_sub
'if min wieght is not meet the return 0's
no_body:
    fsa_cb = 0
    fsa_is = 0
    fsa_bf = 0
    fsa_tf = 0
    fsa_ss = 0

    GoTo end_sub
'if error return default numbers
error_out:
    fsa_cb = 300
    fsa_is = 500
    fsa_bf = 255
    fsa_tf = 100
    fsa_ss = 5
end_sub:
    carray.CBcoefa(cindex) = fsa_cb
    carray.IScoefa(cindex) = fsa_is
    carray.BFcoefa(cindex) = fsa_bf
    carray.TFcoefa(cindex) = fsa_tf
```

```
    carray.SSprofa(cindex) = fsa_ss
    cindex = cindex + 1
    If cindex > MARRAY Then cindex = 1
    Exit Sub
 End Sub
Private Sub Class_Initialize()
    'setup coefficent arrays to 0
    Dim i As Integer
    For i = 1 To MARRAY
        carray.BFcoefa(i) = 0
        carray.CBcoefa(i) = 0
        carray.IScoefa(i) = 0
        carray.SSprofa(i) = 0
        carray.TFcoefa(i) = 0
    Next i
    cindex = 1
End Sub
```

APPENDIX B

```
Option Explicit
Dim PHW As Double
Dim PSW As Double

Dim xlApp As Excel.Application

Dim xlBook As Excel.Workbook

Dim xlSheet As Excel.Worksheet

Dim X1 As Double
Dim X2 As Double
Dim x3 As Double
Dim x4 As Double
Dim x5 As Double
Dim x6 As Double
Dim x7 As Double
Dim x8 As Double
Dim x9 As Double

Dim SpineData As Double
Dim Stopit As Integer
Dim Head As Integer
Dim Feet As Integer

'Indices of last grid index selected
Dim LastRow As Long
Dim LastCol As Long
Const NumHold As Integer = 3
'Indices of the current grid index being dragged
Dim PickRow As Long
Dim PickCol As Long

'Storage for easier reference later
Dim NumRows As Long
Dim NumColumns As Long

'True when rotating, etc
Dim IsModifying As Boolean

'Keeps track of the region the mouse is in
Dim Region As Long
Dim OldRegion As Long

'Keeps track of the current row and column the mouse is on
Dim Row As Long
Dim Col As Long

'ASCII Character constants
Const CharEnter As Integer = 13

'Capture any double-clicks the user does
Dim DoubleClick As Boolean

Public fsa As Long
'Storage for drawing the zoom rectangle
Dim Result As Long
Dim PenHandle As Long
Dim OldPenHandle As Long
Dim ChartDc As Long
```

```
Dim StartPoint As POINTAPI
Dim EndPoint As POINTAPI
Dim MaxRow As Long
Dim MaxCol As Long
Dim Corner1 As POINTAPI
Dim Corner2 As POINTAPI
'Storage for tracking the mouse
Dim px As Long
Dim py As Long
'Storage for the zoom values
Dim XStart As Double
Dim YStart As Double
Dim XEnd As Double
Dim YEnd As Double
Dim StartRow As Long
Dim EndRow As Long
Dim StartCol As Long
Dim EndCol As Long
Dim Val As Double
Dim Distance As Long

'Auxdata window stuff
Dim AuxSensors As Long
Dim AuxSum As Double
Dim AuxWidth As Double
Dim AuxLength As Double
Dim AuxAverage As Double

'Overall window stuff
Dim FSASum As Double
Dim FSAAverage As Double
Dim FSASensors As Long
Dim TorsoAverage As Double
Dim TorsoSensors As Integer
Dim ShoulderAverage As Double
Dim ShoulderWidth As Double
Dim HipAverage As Double
Dim WaistAverage As Double
Dim HipMaxWidth As Double
Dim WAverageWidth As Double
Dim FSAWeight As Double
Dim FSAHeight As Double
Dim FSAIspring As Double
Dim TorsoLength As Double


Private Sub Chart3D1_DblClick()
'Capture the double click.

    DoubleClick = True
End Sub

Private Sub Chart3D1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
'Watch for the user to press the mouse button so we can create the
' data rectangle to use for the zoom process.

    'Make sure it is the left button and then get the needed information
    If Button = 1 And Shift = 0 Then

        Chart3D1.Refresh
        'Get the API information from the main Chart
```

```
        ChartDc = GetDC(Chart3D1.hWnd)
        PenHandle = CreatePen(0, 2, QBColor(0))
        OldPenHandle = SelectObject(ChartDc, PenHandle)
        Result = SetROP2(ChartDc, vbNotXorPen)

        'Get the number of rows and columns in use
        MaxRow = Chart3D1.ChartGroups(1).ElevationData.RowCount
        MaxCol = Chart3D1.ChartGroups(1).ElevationData.ColumnCount

        'Get the pixel co-ordinates of the lower-left and upper-right corners of the
        ' main chart so we can constrain the "Rubber Band" to stay on the data area
        Chart3D1.ChartGroups(1).DataIndexToCoord 1, 1, Corner1.x, Corner1.y
        Chart3D1.ChartGroups(1).DataIndexToCoord MaxRow, MaxCol, Corner2.x, Corner2.y

        px = x / Screen.TwipsPerPixelX                  'Convert the mouse location to pixels
        py = y / Screen.TwipsPerPixelY

        'If we are outside the chart, set the values to be outside the allowable range
        If px < Corner1.x Or px > Corner2.x Then
            StartPoint.x = -1
            StartPoint.y = -1
            EndPoint.x = -1
            EndPoint.y = -1

            'Release the resources as we no longer need them
            Result = SelectObject(ChartDc, OldPenHandle)
            Result = DeleteObject(PenHandle)
            Result = ReleaseDC(Chart3D1.hWnd, ChartDc)
            Exit Sub
        End If
        If py < Corner2.y Or py > Corner1.y Then
            StartPoint.x = -1
            StartPoint.y = -1
            EndPoint.x = -1
            EndPoint.y = -1

            'Release the resources as we no longer need them
            Result = SelectObject(ChartDc, OldPenHandle)
            Result = DeleteObject(PenHandle)
            Result = ReleaseDC(Chart3D1.hWnd, ChartDc)
            Exit Sub
        End If

        'Set the startpoint of the rectangle to the current mouse position
        StartPoint.x = px
        StartPoint.y = py
        EndPoint.x = px
        EndPoint.y = py

        'Draw the "Rubber Band" rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        'Release the resources as we no longer need them
        Result = SelectObject(ChartDc, OldPenHandle)
        Result = DeleteObject(PenHandle)
        Result = ReleaseDC(Chart3D1.hWnd, ChartDc)

        Region = Chart3D1.ChartGroups(1).CoordToDataCoord(StartPoint.x, StartPoint.y, XStart, YS
tart, Val)
        Region = Chart3D1.ChartGroups(1).CoordToDataIndex(px, py, Row, Col, Distance) 'Get the D
ata Values at the current location
```

```
        'Store the row and column values for use in the sub-set creation later
        StartRow = Row
        StartCol = Col

    End If
End Sub
Private Sub Chart3D1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
'Track the movement of the mouse and update the "Rubber Band" rectangle.

    If Button = 1 And StartPoint.x <> -1 And Shift = 0 Then

        'Get the API information from the main Chart
        ChartDc = GetDC(Chart3D1.hWnd)
        PenHandle = CreatePen(0, 2, QBColor(0))
        OldPenHandle = SelectObject(ChartDc, PenHandle)
        Result = SetROP2(ChartDc, vbNotXorPen)

        'Get rid of the old rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        'Convert the screen co-ordinates to pixels
        px = x / Screen.TwipsPerPixelX
        py = y / Screen.TwipsPerPixelY

        'Constrain the "Rubber Band" rectangle to stay on the data area of the chart
        If px >= Corner1.x And px <= Corner2.x Then
            EndPoint.x = px
        Else
            If px < Corner1.x Then
                EndPoint.x = Corner1.x
            Else
                EndPoint.x = Corner2.x
            End If
        End If
        If py >= Corner2.y And py <= Corner1.y Then
            EndPoint.y = py
        Else
            If py > Corner1.y Then
                EndPoint.y = Corner1.y
            Else
                EndPoint.y = Corner2.y
            End If
        End If

        'Draw the new rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        'Release the resources as we no longer need them
        Result = SelectObject(ChartDc, OldPenHandle)
        Result = DeleteObject(PenHandle)
        Result = ReleaseDC(Chart3D1.hWnd, ChartDc)
    End If
End Sub

Private Sub Chart3D1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
'Capture the mouse up event so we know when the user is done creating the rectangle.
'Copy the current graph to one of the empty locations, and then perform the zoom.

    Static i As Integer
    Static J As Integer
```

```
    Dim hd As Integer
    Dim hld As Boolean

    hd = -1

    'Check and make sure there is data to zoom in on first and exit if there isn't any selected
    If Abs(StartPoint.x - EndPoint.x) < 1 Or Abs(StartPoint.y - EndPoint.y) < 1 Then

        'Get the API information from the main Chart
        ChartDc = GetDC(Chart3D1.hWnd)
        PenHandle = CreatePen(0, 2, QBColor(0))
        OldPenHandle = SelectObject(ChartDc, PenHandle)
        Result = SetROP2(ChartDc, vbNotXorPen)

        'Clear the rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        'Release the resources as we no longer need them
        Result = SelectObject(ChartDc, OldPenHandle)
        Result = DeleteObject(PenHandle)
        Result = ReleaseDC(Chart3D1.hWnd, ChartDc)
        Exit Sub
    End If

    If Button = 1 And StartPoint.x <> -1 Then

        'Get the API information from the main Chart
        ChartDc = GetDC(Chart3D1.hWnd)
        PenHandle = CreatePen(0, 2, QBColor(0))
        OldPenHandle = SelectObject(ChartDc, PenHandle)
        Result = SetROP_(ChartDc, vbNotXorPen)

        'Get rid of the old rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        px = x / Screen.TwipsPerPixelX              'Convert screen co-ordinates to pixels
        py = y / Screen.TwipsPerPixelY

        'Constrain the "Rubber Band" rectangle to the data area of the chart
        If px >= Corner1.x And px <= Corner2.x Then
            EndPoint.x = px
        Else
            If px < Corner1.x Then
                EndPoint.x = Corner1.x
            Else
                EndPoint.x = Corner2.x
            End If
        End If
        If py >= Corner2.y And py <= Corner1.y Then
            EndPoint.y = py
        Else
            If py > Corner1.y Then
                EndPoint.y = Corner1.y
            Else
                EndPoint.y = Corner2.y
            End If
        End If

        'Draw the new rectangle
        Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)
```

```
        'Capture values for use in the creation of the subset later
        Region = Chart3D1.ChartGroups(1).CoordToDataCoord(EndPoint.x, EndPoint.y, XEnd, YEnd, Va
l)
        Region = Chart3D1.ChartGroups(1).CoordToDataIndex(EndPoint.x, EndPoint.y, Row, Col, Dist
ance) 'Get the Data Values at the current location

        If Region = oc3dRegionInChartArea Then
            EndRow = Row
            EndCol = Col
        End If


        'Clear the rectangle
'       Result = Rectangle(ChartDc, StartPoint.x, StartPoint.y, EndPoint.x, EndPoint.y)

        'Release the resources as we no longer need them
        Result = SelectObject(ChartDc, OldPenHandle)
        Result = DeleteObject(PenHandle)
        Result = ReleaseDC(Chart3D1.hWnd, ChartDc)

        Debug.Print StartRow; StartCol; EndRow; EndCol

        'Switch around the rows and cols to make things easier
        If StartRow > EndRow Then
            i = StartRow
            StartRow = EndRow
            EndRow = i
        End If

        If StartCol > EndCol Then
            i = StartCol
            StartCol = EndCol
            EndCol = i
        End If

        AuxSensors = 0
        AuxSum = 0

        For i = StartCol To EndCol
            For J = StartRow To EndRow
                If Chart3D1.ChartGroups(1).ElevationData.Value(J, i) > 0 Then
                    AuxSum = AuxSum + Chart3D1.ChartGroups(1).ElevationData.Value(J, i)
                    AuxSensors = AuxSensors + 1
                End If
            Next J
        Next i

        If Len(CommonDialog1.filename) > 0 Then
            AuxWidth = Abs(YEnd - YStart)
            AuxLength = Abs(XEnd - XStart)
            AuxAverage = AuxSum / AuxSensors


            'Reset the location of the highlighted area

        End If
    End If
End Sub


Private Sub Command1_Click()
```

```
    If Me.BFcoef = 0 Or Me.CBcoef = 0 Or Me.IScoef = 0 Or Me.TFcoef = 0 Then
        EntryError.ErrorText.Caption = "You must have a FSA file or manually entered data to pro
ceed"
        EntryError.Show 1
        Exit Sub
    End If


    Main.Show 1
End Sub


Private Sub Command2_Click()
    GoExcel
End Sub

Private Sub Form_Load()
'This is where it all begins!

    Frame2.Enabled = False

    'Start with the form in the top-left corner
    Me.Top = 50
    Me.Left = 50

    'Setup the need variables
    Dim r As Integer
    Dim c As Integer
    Dim AxisValue As Double
    Dim delta As Double

    'These are backwards so because the grid is the main problem
    ' to contend with due to all the inversion necessary
    NumRows = Chart3D1.ChartGroups(1).ElevationData.ColumnCount
    NumColumns = Chart3D1.ChartGroups(1).ElevationData.RowCount

    'Set a default value
    DoubleClick = False

    'This puts the values of the Grid Index points into the header row
    ' of the grid control.  (grid control is at the bottom of the window)
    'NOTE: Becuase of the rotation of the graph, the columns and rows are
    '      reversed in order to fill the grid in correspondance with the graph
    delta = Chart3D1.ChartGroups(1).ElevationData.RowDelta(1)
    AxisValue = Chart3D1.ChartGroups(1).ElevationData.RowOrigin

    'This puts the values of the Grid Index points into the header column
    ' of the grid control. (grid control is at the bottom of the window)
    'NOTE: Becuase of the rotation of the graph, the columns and rows are
    '      reversed in order to fill the grid in correspondance with the
    '      chart
    delta = Chart3D1.ChartGroups(1).ElevationData.ColumnDelta(1)
    AxisValue = Chart3D1.ChartGroups(1).ElevationData.ColumnOrigin

    'Change the color of the Grid Lines
    Chart3D1.ChartArea.Axes("X").MajorGrid.Style.Color = ocColorCornflowerBlue
    Chart3D1.ChartArea.Axes("Y").MajorGrid.Style.Color = ocColorCornflowerBlue
    Chart3D1.ChartArea.Axes("Z").MajorGrid.Style.Color = ocColorCornflowerBlue
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)
'End the program.

    End
End Sub



Private Sub mnuAbout_Click()
'User wants to see what to do in this demo.

    With CommonDialog1
        .HelpCommand = cdlHelpContext
        .HelpContext = 18
        .HelpFile = App.HelpFile
        .ShowHelp
    End With
End Sub

Private Sub mnuAboutOlectra_Click()
'User wants to see what Olectra Chart 3D is all about.

    With CommonDialog1
        .HelpCommand = cdlHelpContext
        .HelpContext = 19
        .HelpFile = App.HelpFile
        .ShowHelp
    End With
End Sub

Private Sub mnuExit_Click()
'Exit the program.

    Unload Me
End Sub



Private Sub mnuOpen_Click()
    Dim sFile As String

    With CommonDialog1
        .filename = ""
        .Flags = 0
        'To Do
        'set the flags and attributes of the
        'common dialog control
        .Filter = "FSA Files (*.FSA)|*.*"
        .ShowOpen
        If Len(.filename) = 0 Then
            Exit Sub
        End If
        sFile = .filename
    End With

'    cancel = CommonDialog1.Action
    Crunchit (sFile)
```

```
End Sub
Private Sub Crunchit(sFile As String)

    Dim r As Integer
    Dim c As Integer
    Dim RAv As Double
    Dim avrav As Double
    Dim InARow As Integer
    Dim TorsoBottom As Integer
    Dim TorsoTop As Integer
    Dim TorsoLeft As Integer
    Dim TorsoRight As Integer
    Dim Brow As Integer
    Dim HipSum As Double
    Dim WaistSum As Double
    Dim HipSensors As Integer
    Dim WFirst As Integer
    Dim WLast As Integer
    Dim LastInARow As Integer
    Dim ShoulderSum As Double
    Dim ShoulderSensors As Integer
    Dim SLast As Integer
    Dim SFirst As Integer
    Dim Stretch As Integer
    Dim WaistSensors As Integer
    Dim DTemp As Double
    Dim TorsoCenter As Double
    Dim test As Variant

    ReadFile (sFile)

    Chart3D1.IsBatched = True

    fsa = 0
    FSASum = 0
    FSASensors = 0

    'fill in the chart region - batched

    For c = 1 To NumColumns
        For r = 1 To NumRows

            DTemp = VistaFile.FSAData(fsa) * VistaFile.UnitMultiplier

            Oex2000.Put_FSAData VistaFile.FSAData(fsa), fsa

            Chart3D1.ChartGroups(1).ElevationData.Value(c, r) = DTemp

            fsa = fsa + 1
        Next r

    Next c

    Chart3D1.IsBatched = False

    BFcoef.Text = Oex2000.BFcoef

    CBcoef.Text = Oex2000.CBcoef

    TFcoef.Text = Oex2000.TFcoef
```

```
    IScoef.Text = Oex2000.IScoef

    SSprof.Text = Oex2000.SSprof

    Dim holder1, holder2, holder3 As Double

    If Option1(0).Value = True Then

        HeadZone.Text = Format(Oex2000.FSAWeight * 0.11, "00")

        holder1 = (Oex2000.FSAWeight * 0.55)

        holder2 = Oex2000.ShoulderAverage + Oex2000.WaistAverage + Oex2000.HipAverage

        holder3 = 100 - holder2

        If holder3 < 0 Then holder3 = -holder3

        holder3 = holder3 / 2

        ShoulderZone.Text = Format(holder1 * ((Oex2000.ShoulderAverage + holder3) / 100), "00")

        WaistZone.Text = Format(holder1 * ((Oex2000.WaistAverage) / 100), "00")

        HipZone.Text = Format(holder1 * ((Oex2000.HipAverage + holder3) / 100), "00")

        ThighZone.Text = Format(Oex2000.FSAWeight * 0.2, "00")

        FeetZone.Text = Format(Oex2000.FSAWeight * 0.1, "00")


    Else




    End If

    frmSurface.Caption = "L&P Controls FSA Statistic Tool - " + sFile

End Sub

Private Sub Option1_Click(index As Integer)

    If index = 1 Then Frame2.Enabled = True Else: Frame2.Enabled = False


End Sub

Private Sub Utility_Click()
    Main.Show 1


End Sub
```

```
Public Sub GoExcel()
    Set xlApp = CreateObject("Excel.Application")

    Set xlBook = xlApp.Workbooks.Add

    Set xlSheet = xlBook.Worksheets(1)

    xlApp.Visible = True

    xlSheet.Cells(1, 1) = "Subject"
    xlSheet.Cells(1, 2) = "CB"
    xlSheet.Cells(1, 3) = "IS"
    xlSheet.Cells(1, 4) = "BF"
    xlSheet.Cells(1, 5) = "TF"

    Dim sFile As String
    Dim i As Integer


     For i = 1 To 79
        If i < 10 Then
        sFile = "C:\FSADATA\0" & i & ".fsa"
        Else: sFile = "C:\FSADATA\" & i & ".fsa"
        End If
        Crunchit (sFile)
        xlSheet.Cells(i + 1, 1) = i
        xlSheet.Cells(i + 1, 2) = Format$(Oex2000.CBcoef, "0")
        xlSheet.Cells(i + 1, 3) = Format$(Oex2000.IScoef, "0")
        xlSheet.Cells(i + 1, 4) = Format$(Oex2000.BFcoef, "0")
        xlSheet.Cells(i + 1, 5) = Format$(Oex2000.TFcoef, "0")

    Next i

    xlSheet.SaveAs ("FSAconversion")

    xlApp.Quit

End Sub
```

Home - 1

```
Private Done As Integer

Private Sub Form_Load()

    Done = False
    DoEvents
    Call ApiHome

    Done = True

End Sub

Private Sub Timer1_Timer()

    Dim Complete As Long

   If ProgressBar1.Value = 100 Then

        ProgressBar1.Value = 0
        Timer1.Enabled = False
        Complete = ApiComplete(0, "HOME")
        If Complete = &HAABFF Then
            Status.Caption = "Complete"
            Unload Me
            Exit Sub
        End If

        Timer1.Enabled = True

   End If

   ProgressBar1.Value = ProgressBar1.Value + 1
End Sub
```

```
Option Explicit

Private Sub CancelButton_Click()
    Main.Choice = 1
    Unload Me

End Sub

Private Sub OKButton_Click()
    Main.Choice = 0
    Unload Me

End Sub
```

```
Public Project As Variant
Public ProjectDate As Variant
Public Choice As Integer
Public Head As Variant
Public Trunk As Variant
Public Thighs As Variant
Public Legs As Variant
Public Feet As Variant
Public TestStage As Variant
Public MonitorStage As Variant

Private Type Oexrecord
    ' identity stuff
    Description As String
    pDate As String
    Setup As Integer
    ' coefficients
    BFcoef As Double
    CBcoef As Double
    TFcoef As Double
    IScoef As Double
    SSprof As Double
    ' zone stuff
    Hdzone As Double
    Szone As Double
    Wzone As Double
    Hpzone As Double
    Tzone As Double
    Fzone As Double
    ' positional feedback
    Position(1 To 20) As Single
    ' calibration stuff
    Calibration(1 To 20) As Single
End Type

Private oexdata As Oexrecord


' Home Button
Private Sub Command1_Click()

    Home.Show 1

End Sub
' Calibrate Button
Private Sub Command3_Click()

    Calibrate.Show 1

End Sub
' Reset Button
Private Sub Command4_Click()

    Reset.Show 1

End Sub



' Test Button
Private Sub Command5_Click()
```

```
'      '
    Dim filename As Variant
    Dim i As Integer

    If ProjectText.Text = "" Then

        EntryError.ErrorText.Caption = "No Project Entered"
        EntryError.Show 1
        Exit Sub

    End If

    ' build the filename with directory and extension

    filename = "\oex2000\" + ProjectText.Text + ".oex"

    ' verify the existance or lack thereof

    If Dir(filename) = "" Then


    Else

        ' file already exists - replace = 0, cancel = 1
        FileError.Show 1

        If Main.Choice = 1 Then Exit Sub

    End If

    ' if we get here we have an open file "filename"
    ' and the data has been justified enough to proceed with the test

    If Option1(0).Value = True Then

        ApiHeadZone frmSurface.HeadZone.Text * 0.0151
        ApiShoulderZone (frmSurface.ShoulderZone.Text * 0.0151) / 3
        ApiWaistZone (frmSurface.WaistZone.Text * 0.0151) / 4
        ApiHipZone (frmSurface.HipZone.Text * 0.0151) / 2
        ApiThighZone (frmSurface.ThighZone.Text * 0.0151) / 2
        ApiFeetZone (frmSurface.FeetZone.Text * 0.0151) / 2

    End If

    Timer1.Enabled = True

    TestStage = 1
    MonitorStage = 1


   ' retract to zero start - home if not initialized
    Status.Caption = "Initializing..."
    DoEvents
    Call ApiRetract

wait_Retract:

    DoEvents

    If ApiComplete(0, "RETRACT") <> &HAABFF Then GoTo wait_Retract
```

```
    Timer1.Enabled = False

    Instruct1.Show 1

    If Main.Choice = 1 Then Exit Sub

    Timer1.Enabled = True

    ' run the product test now that everytihing is setup
    Status.Caption = "Testing Product..."
    Call ApiTest

wait_Test:

    DoEvents

    If ApiComplete(0, "TEST") <> &HAABFF Then GoTo wait_Test

    ' write test data to the open file and complete

    Status.Caption = "Storing Test Data..."

    oexdata.Description = Description.Text
    oexdata.pDate = DateText.Text
    ' only valid one right now
    oexdata.Setup = 0

    oexdata.BFcoef = BFcoef.Text
    oexdata.CBcoef = CBcoef.Text
    oexdata.TFcoef = TFcoef.Text
    oexdata.IScoef = IScoef.Text
    oexdata.SSprof = SSprof.Text

    oexdata.Hdzone = frmSurface.HeadZone.Text
    oexdata.Szone = frmSurface.ShoulderZone.Text
    oexdata.Wzone = frmSurface.WaistZone.Text
    oexdata.Hpzone = frmSurface.HipZone.Text
    oexdata.Tzone = frmSurface.ThighZone.Text
    oexdata.Fzone = frmSurface.FeetZone.Text

    For i = 1 To 10
        oexdata.Position(i) = Axis.FloatValueOf(i, "PFPOS")
        oexdata.Calibration(i) = Axis.FloatValueOf(i, "CAL")
    Next i

    For i = 12 To 20 Step 2
        oexdata.Position(i) = Axis.FloatValueOf(i, "PFPOS")
        oexdata.Calibration(i) = Axis.FloatValueOf(i, "CAL")
    Next i

    'close file after writing all project info

    Open filename For Binary Access Write As #1

    Put #1, 1, oexdata

    Close #1

    Status.Caption = "Test Complete!"
    Timer1.Enabled = False
    ProgressBar1.Value = 100
```

```
'      '
End Sub

Private Sub Command7_Click()

    Calibrate.Show 1

End Sub

Private Sub Command6_Click()

    EStop.Show 1

End Sub

Private Sub Command8_Click()

    SetDate.Show 1

End Sub

Private Sub Form_Load()

    ApiOpenPort

    DateText.Text = SetDate.Calendar1.Value

    'combobox is disabled until support can be written
    Combo1.Enabled = False

     BFcoef.Text = frmSurface.BFcoef.Text
    CBcoef.Text = frmSurface.CBcoef.Text
    TFcoef.Text = frmSurface.TFcoef.Text
    IScoef.Text = frmSurface.IScoef.Text
    SSprof.Text = frmSurface.SSprof.Text

    Option1(1).Enabled = False
    Option1(2).Enabled = False
    Option3(1).Enabled = False


End Sub

Private Sub Form_Unload(Cancel As Integer)

    ApiClosePort
    Unload SetDate

End Sub

Private Sub Option1_Click(index As Integer)


    ApiSetDrives (index)
    If Option1(2).Value = True Then
        Option1(2).Value = False
        Option1(0).Value = True
        EntryError.ErrorText.Caption = "Option not yet supported"
        EntryError.Show 1
    End If
```

```
End Sub

Private Sub Retract_Click()

Position.Show 1



End Sub

Private Sub Summary_Click()
    sum.Show 1
End Sub

Private Sub Timer1_Timer()

    If TestStage = 0 Then

    ElseIf TestStage = 1 Then

    ElseIf TestStage = 2 Then

    ElseIf TestStage = 3 Then

    End If

     If ProgressBar1.Value = 100 Then
        ProgressBar1.Value = 0
     End If
    ProgressBar1.Value = ProgressBar1.Value + 1
End Sub
```

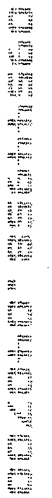
```
Reset - 1

Private Done As Integer

Private Sub Form_Load()

    Done = False

    DoEvents

    Call ApiReset

    Done = True

End Sub

Private Sub Timer1_Timer()

    Dim Complete As Long

   If ProgressBar1.Value = 100 Then

        ProgressBar1.Value = 0
        Timer1.Enabled = False

        If Done = True Then
            Status.Caption = "Complete"
            Unload Me
            Exit Sub

        End If

        Timer1.Enabled = True

   End If

   ProgressBar1.Value = ProgressBar1.Value + 1
End Sub
```

```
SetDate - 1

Option Explicit

Private Sub CancelButton_Click()

    Unload Me

End Sub

Private Sub Form_Load()

    Calendar1.Today

End Sub

Private Sub OKButton_Click()

    Main.ProjectDate = Calendar1.Value
    Main.DateText.Text = Main.ProjectDate
    Unload Me

End Sub
```

```
sum - 1

Option Explicit


Private Type Oexrecord
    ' identity stuff
    Description As String
    pDate As String
    Setup As Integer
    ' coefficients
    BFcoef As Double
    CBcoef As Double
    TFcoef As Double
    IScoef As Double
    SSprof As Double
    ' zone stuff
    Hdzone As Double
    Szone As Double
    Wzone As Double
    Hpzone As Double
    Tzone As Double
    Fzone As Double
    ' positional feedback
    Position(1 To 20) As Single
    ' calibration stuff
    Calibration(1 To 20) As Single
End Type


Private Sub mnuAbout_Click()
'User wants to see what to do in this demo.

    With CommonDialog1
        .HelpCommand = cdlHelpContext
        .HelpContext = 18
        .HelpFile = App.HelpFile
        .ShowHelp
    End With
End Sub

Private Sub mnuAboutOlectra_Click()
'User wants to see what Olectra Chart 3D is all about.

    With CommonDialog1
        .HelpCommand = cdlHelpContext
        .HelpContext = 19
        .HelpFile = App.HelpFile
        .ShowHelp
    End With
End Sub

Private Sub mnuExit_Click()
'Exit the program.

    Unload Me
End Sub



Private Sub mnuOpen_Click()
    Dim sFile As String
```

```
    With CommonDialog1
        .filename = ""
        .Flags = 0
        'To Do
        'set the flags and attributes of the
        'common dialog control
        .Filter = "Oex2000 Files (*.OEX)|*.*"
        .ShowOpen
        If Len(.filename) = 0 Then
            Exit Sub
        End If
        sFile = .filename
    End With

'    cancel = CommonDialog1.Action
    Crunchit (sFile)

End Sub
Private Sub Crunchit(sFile As String)

    Dim oexdata As Oexrecord

    Dim i As Integer

    Open sFile For Binary Access Read As #1

    Get #1, 1, oexdata

    Close #1

    Text3.Text = sFile

    Text2.Text = oexdata.Description

    BFcoef.Text = Oex2000.BFcoef

    CBcoef.Text = Oex2000.CBcoef

    TFcoef.Text = Oex2000.TFcoef

    IScoef.Text = Oex2000.IScoef

    SSprof.Text = Oex2000.SSprof


    HeadZone.Text = oexdata.Hdzone

    ShoulderZone.Text = oexdata.Szone

    WaistZone.Text = oexdata.Wzone

    HipZone.Text = oexdata.Hpzone

    ThighZone.Text = oexdata.Tzone

    FeetZone.Text = oexdata.Fzone

    frmSurface.Caption = "L&P Controls FSA Statistic Tool - " + sFile

    For i = 1 To 10
```

```
sum - 3

'       'Text1(i - 1).Text = oexdata.Position(i)
    Next i
    Text1(10).Text = oexdata.Position(12)
    Text1(11).Text = oexdata.Position(14)
    Text1(12).Text = oexdata.Position(16)
    Text1(13).Text = oexdata.Position(18)
    Text1(14).Text = oexdata.Position(20)




End Sub

Private Sub Option1_Click(index As Integer)

    If index = 1 Then Frame2.Enabled = True Else: Frame2.Enabled = False


End Sub
```



```
AxisControl - 1

'
' data acquisition and motion control program
'

Public Axis As New ISP

Const All = 255

Private Status As Long

' holds the trace results
Dim x(250), y(2, 250)

' number of times to retry communications
Const Retrys = 5

' windows sleep function
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Public Sub ApiHeadZone(Setting As String)

    Axis.ChangeValue 1, "SETTING", Setting

End Sub
Public Sub ApiShoulderZone(Setting As String)

    Axis.ChangeValue 2, "SETTING", Setting
    Axis.ChangeValue 3, "SETTING", Setting
    Axis.ChangeValue 4, "SETTING", Setting

End Sub
Public Sub ApiWaistZone(Setting As String)

    Axis.ChangeValue 5, "SETTING", Setting

End Sub
Public Sub ApiHipZone(Setting As String)

    Axis.ChangeValue 6, "SETTING", Setting
    Axis.ChangeValue 7, "SETTING", Setting
    Axis.ChangeValue 8, "SETTING", Setting
    Axis.ChangeValue 9, "SETTING", Setting

End Sub
Public Sub ApiThighZone(Setting As String)


    Axis.ChangeValue 10, "SETTING", Setting

    Axis.ChangeValue 12, "SETTING", Setting

End Sub
Public Sub ApiFeetZone(Setting As String)

    Axis.ChangeValue 14, "SETTING", Setting
    Axis.ChangeValue 16, "SETTING", Setting
    Axis.ChangeValue 18, "SETTING", Setting
    Axis.ChangeValue 20, "SETTING", Setting

End Sub
```

```
Public Sub ApiHome()

    Axis.ChangeValue All, "HOME", "1.0"

End Sub

Public Sub ApiProduct()

    Axis.ChangeValue All, "PRODUCT", "1.0"

End Sub

Public Sub ApiTest()

    Axis.ChangeValue All, "TEST", "1.0"

End Sub

Public Sub ApiRetract()

    Axis.ChangeValue All, "RETRACT", "1.0"

End Sub

Public Sub ApiCalibrate()

    Axis.ChangeValue All, "CALIBRATE", "1.0"

End Sub

Public Sub ApiReset()

    Axis.ChangeValue All, "SWE", "0"

    Axis.PgmStop All
    Axis.Reset All
    Axis.ClearPgm (All)
    Axis.LoadPgm All, "MAIN"
    Axis.PgmRun All

    Call ApiSetDrives(0)

End Sub

Public Sub ApiEstop()

    Axis.EStop All
        DoEvents
    Axis.Reset All
        DoEvents
    Axis.ClearPgm All
        DoEvents
    Axis.LoadPgm All, "MAIN"
        DoEvents
    Axis.PgmRun All
        DoEvents


End Sub

Public Sub ApiSetDrives(Setup As Integer)
```

AxisControl - 3

```
    ' which axis to deal with

    If Setup = 0 Then

        Axis.ChangeValue 1, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 2, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 3, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 4, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 5, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 6, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 7, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 8, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 9, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 10, "SWE", "1.0"
        DoEvents

        Axis.ChangeValue 12, "SWE", "1.0"

        DoEvents
        Axis.ChangeValue 14, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 16, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 18, "SWE", "1.0"
        DoEvents
        Axis.ChangeValue 20, "SWE", "1.0"

        'Axis.ChangeValue 13, "SWE", "0.0"
        'Axis.ChangeValue 15, "SWE", "0.0"
        'Axis.ChangeValue 17, "SWE", "0.0"
        'Axis.ChangeValue 19, "SWE", "0.0"

    ElseIf Setup = 1 Then

'       Axis.ChangeValue 1, "SWE", "1.0"
'       Axis.ChangeValue 2, "SWE", "1.0"
'       Axis.ChangeValue 3, "SWE", "1.0"
'       Axis.ChangeValue 4, "SWE", "1.0"
'       Axis.ChangeValue 5, "SWE", "1.0"
'       Axis.ChangeValue 6, "SWE", "1.0"
'       Axis.ChangeValue 7, "SWE", "1.0"
'       Axis.ChangeValue 8, "SWE", "1.0"
'       Axis.ChangeValue 9, "SWE", "1.0"
'       Axis.ChangeValue 10, "SWE", "1.0"
'       Axis.ChangeValue 11, "SWE", "1.0"
'       Axis.ChangeValue 12, "SWE", "1.0"

'       Axis.ChangeValue 14, "SWE", "1.0"
'       Axis.ChangeValue 16, "SWE", "1.0"
'       Axis.ChangeValue 18, "SWE", "1.0"
'       Axis.ChangeValue 20, "SWE", "1.0"
```

```
'       Axis.ChangeValue 13, "SWE", "1.0"
'       Axis.ChangeValue 15, "SWE", "1.0"
'       Axis.ChangeValue 17, "SWE", "1.0"
'       Axis.ChangeValue 19, "SWE", "1.0"


    End If


End Sub

Public Function ApiComplete(Setup As Integer, Task As String) As Long
    Status = 0
    If Setup = 0 Then

        If Axis.FloatValueOf(1, Task) = -1 Then Status = Status Or 1
        DoEvents
        If Axis.FloatValueOf(2, Task) = -1 Then Status = Status Or 2
        DoEvents
        If Axis.FloatValueOf(3, Task) = -1 Then Status = Status Or 4
        DoEvents
        If Axis.FloatValueOf(4, Task) = -1 Then Status = Status Or 8
        DoEvents
        If Axis.FloatValueOf(5, Task) = -1 Then Status = Status Or 16
        DoEvents
        If Axis.FloatValueOf(6, Task) = -1 Then Status = Status Or 32
        DoEvents
        If Axis.FloatValueOf(7, Task) = -1 Then Status = Status Or 64
        DoEvents
        If Axis.FloatValueOf(8, Task) = -1 Then Status = Status Or 128
        DoEvents
        If Axis.FloatValueOf(9, Task) = -1 Then Status = Status Or 256
        DoEvents
        If Axis.FloatValueOf(10, Task) = -1 Then Status = Status Or 512
        DoEvents

        If Axis.FloatValueOf(12, Task) = -1 Then Status = Status Or 2048
'        If Axis.FloatValueOf(13, Task) = -1 Then Home = Home Or 4096
        DoEvents
        If Axis.FloatValueOf(14, Task) = -1 Then Status = Status Or 8192
'        If Axis.FloatValueOf(15, Task) = -1 Then Home = Home Or 16384
        DoEvents
        If Axis.FloatValueOf(16, Task) = -1 Then Status = Status + 32768
'        If Axis.FloatValueOf(17, Task) = -1 Then Home = Home Or 65536
        DoEvents
        If Axis.FloatValueOf(18, Task) = -1 Then Status = Status + &H20000
'        If Axis.FloatValueOf(19, Task) = -1 Then Home = Home Or &H40000
        DoEvents
        If Axis.FloatValueOf(20, Task) = -1 Then Status = Status + &H80000

    ElseIf Setup = 1 Then

    End If

    ApiComplete = Status

End Function
Public Function ApiStartPos(Setup As Integer) As Single

    Dim fbpos(1 To 20) As Single
```

```
    Dim tr As Single
    Dim i As Integer

    If Setup = 0 Then

        For i = 1 To 20
            fbpos(i) = 0
        Next i

        For i = 1 To 10
            fbpos(i) = Axis.FloatValueOf(i, "FPOS")
            DoEvents
        Next i

        For i = 12 To 20 Step 2
            fbpos(i) = Axis.FloatValueOf(i, "FPOS")
            DoEvents
        Next i

    ElseIf Setup = 1 Then

    End If


    ' get the translation ratio
    tr = Axis.FloatValueOf(1, "TR")

    ' signal div zero error
    If tr = 0 Then


    End If

    ' translate the fbpos's to inches
    For i = 1 To 20
        fbpos(i) = fbpos(i) / tr
    Next i
```

```
End Function

Public Sub ApiOpenPort()

    Axis.OpenPort 1

    ApiReset

End Sub

Public Sub ApiClosePort()

    Axis.ClosePort
```

```
End Sub


Public Sub Capture()

    ' holds the results
    Dim x(250), y(2, 250)

    Dim Axis As Integer
    Dim jup As Integer
    Dim jlow As Integer

    Dim ct As Integer
    Dim ys As Single
    Dim ymin As Single

    ' desired axis number
    Axis = 1

    ' what to grab (jup = first set of data, jlow = second set of data)
    '
    ' 0 = analog input (ADC0)
    ' 1 = target position (TPOS)
    ' 2 = target velocity
    ' 3 = target accel
    ' 4 = feedback position
    ' 5 = feedback velocity
    ' 6 = position error
    ' 7 = current reference
    ' 8 = velocity error
    jup = 4
    jlow = 7

    ' total capture time in tenths of a second
    ' with ci_desired = 10, capture of 1 second total time
    ci_desired = 100

    ' whether (=1) or not (=0) to wait for the "WAIT FOR TRIGGER" step within
    ' a program
    trig = 1

    ' capture the data
    GoCap Axis, (ci_desired), (jup), (jlow), (trig)

    ' wait until the data is acquired by the drive
    While Not FinishedCap(Axis)
        DoEvents
    Wend

    ' first set of data (250 pts) into y(1,i)

    a = Cap(Axis, 0)
   ' a check should be made here to make sure len(a) = 250
   ' if not, the data did not make it over...

    Yscale Axis, 0, ct, ys, ymin
    If ys = 0 Then ys = 1

    For i = 1 To 250
        x(i) = 0.001 * (i - 1) * ci_desired / 2.5
```

```
        yyy = Asc(Mid(a, i, 1))
        y(1, i) = yyy / ys + ymin
    Next i

    ' second set of data (250 pts) into y(2,i)

    a = Cap(Axis, 1)
   ' a check should be made here to make sure len(a) = 250
   ' if not, the data did not make it over...

    Yscale Axis, 1, ct, ys, ymin
    If ys = 0 Then ys = 1

    For i = 1 To 250
        yyy = Asc(Mid(a, i, 1))
        y(2, i) = yyy / ys + ymin
    Next i

    ' done

    Stop

End Sub

Public Function GoCap(id As Integer, ci As Integer, ct As Integer, ct2 As Integer, trig As Integ
er) As Boolean

   Dim i As Integer
   Dim a As String

   For i = 1 To Retrys

   If (trig = 0) Then
       Axis.SendPacket (id), Chr(11) + Chr(ci) + Chr(ct) + Chr(ct2)
   Else
       Axis.SendPacket (id), Chr(24) + Chr(ci) + Chr(ct) + Chr(ct2)
   End If

    a = Axis.GetPacket(1)

    If a <> "" Then
         GoCap = True
        Exit Function
    End If

    Next i

    GoCap = False

End Function

Function FinishedCap(id As Integer) As Boolean

    FinishedCap = False
    If Axis.Status(id) And 256 Then FinishedCap = True

End Function

Public Function Cap(id As Integer, thetype As Integer) As String

    Dim i As Integer
```

```
    For i = 1 To Retrys

    Axis.SendPacket (id), Chr(13) + Chr(thetype)
    Sleep 100

    Cap = Axis.GetPacket(250)
    If Cap <> "" Then
        Exit Function
    End If

    Next i

    Cap = ""

End Function

Public Function Yscale(id, which, ByRef captype As Integer, ByRef ys As Single, ByRef ym As Sing
le) As Boolean

    Dim i As Integer
    Dim a As String

    Yscale = False
    ys = 1
    ym = 0

    For i = 1 To Retrys

    Axis.SendPacket (id), Chr(12) + Chr(which)
    Sleep 50
    a = Axis.GetPacket(0)

    If a <> "" Then
        captype = Asc(Left(a, 1)) - 1
        a = Mid(a, 2)
        i = InStr(a, ",")
        If i <> 0 Then
            Yscale = True
            ys = Val(Left(a, i - 1))
            ym = Val(Mid(a, i + 1))
        End If
        Yscale = True
        Exit Function
    End If

    Next i

End Function
```

```
'*************************************************************************
'*
'* Copyright (c) 1998, KL GROUP INC.  All Rights Reserved.
'* http://www.klg.com
'*
'* This file is provided for demonstration and educational uses only.
'* Permission to use, copy, modify and distribute this file for
'* any purpose and without fee is hereby granted, provided that the
'* above copyright notice and this permission notice appear in all
'* copies, and that the name of KL Group not be used in advertising
'* or publicity pertaining to this material without the specific,
'* prior written permission of an authorized representative of
'* KL Group.
'*
'* KL GROUP MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY
'* OF THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
'* TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
'* PURPOSE, OR NON-INFRINGEMENT. KL GROUP SHALL NOT BE LIABLE FOR ANY
'* DAMAGES SUFFERED BY USERS AS A RESULT OF USING, MODIFYING OR
'* DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
'*
'*************************************************************************

'This file contains a list of colors that can be used
' in any VB program.  These colors can also be found in
' the property pages of the Olectra Chart controls.

'The HEX values can also be broken down into their
' Red Green Blue (RGB) equivalents by breaking the number
' down by pairs of digits.  Here is an example:
'    ocColorTurquoise = &HD0E040
'    Red = &HD0 (208), Green = &HE0 (224), Blue = &H40 (64)

'ocDefaultColor is the same as "(Automatic)" in the property pages
Public Const ocDefaultColor As Long = &HFFFF

Public Const ocColorAliceBlue As Long = &HFFF8F0
Public Const ocColorAntiqueWhite As Long = &HD7EBFA
Public Const ocColorAquamarine As Long = &HD4FF7F
Public Const ocColorAzure As Long = &HFFFFF0
Public Const ocColorBeige As Long = &HDCF5F5
Public Const ocColorBisque As Long = &HC4E4FF
Public Const ocColorBlack As Long = &H0
Public Const ocColorBlanchedAlmond As Long = &HCDEBFF
Public Const ocColorBlue As Long = &HFF0000
Public Const ocColorBlueViolet As Long = &HE22B8A
Public Const ocColorBrown As Long = &H2A2AA5
Public Const ocColorBurlywood As Long = &H87B8DE
Public Const ocColorCadetBlue As Long = &HA09E5F

'ocColorChartreuse As Long = &H00FF7F
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorChartreuse As Long = 65407

Public Const ocColorChocolate As Long = &H1E69D2
Public Const ocColorCoral As Long = &H507FFF
Public Const ocColorCornflowerBlue As Long = &HED9564
Public Const ocColorCornsilk As Long = &HDCF8FF
Public Const ocColorCyan As Long = &HFFFF00
Public Const ocColorDarkGoldenrod As Long = &HB86B8
```

```
'ocColorDarkGreen As Long = &H006400
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorDarkGreen As Long = 25600

Public Const ocColorDarkKhaki As Long = &H6BB7BD
Public Const ocColorDarkOliveGreen As Long = &H2F6B55
Public Const ocColorDarkOrange As Long = &H8CFF
Public Const ocColorDarkOrchid As Long = &HCC3299
Public Const ocColorDarkSalmon As Long = &H7A96E9
Public Const ocColorDarkSeaGreen As Long = &H8FBC8F
Public Const ocColorDarkSlateBlue As Long = &H8B3D48
Public Const ocColorDarkSlateGray As Long = &H4F4F2F
Public Const ocColorDarkTurquoise As Long = &HD1CE00
Public Const ocColorDarkViolet As Long = &HD30094
Public Const ocColorDeepPink As Long = &H9314FF
Public Const ocColorDeepSkyBlue As Long = &HFFBF00
Public Const ocColorDodgerBlue As Long = &HFF901E
Public Const ocColorFirebrick As Long = &H2222B2
Public Const ocColorFloralWhite As Long = &HF0FAFF
Public Const ocColorForestGreen As Long = &H228B22
Public Const ocColorGainsboro As Long = &HDCDCDC
Public Const ocColorGhostWhite As Long = &HFFF8F8

'ocColorGold As Long = &H00D7FF
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorGold As Long = 55295

Public Const ocColorGoldenrod As Long = &H20A5DA
Public Const ocColorGray As Long = &HBEBEBE
Public Const ocColorGray0 As Long = &H0
Public Const ocColorGray1 As Long = &H30303
Public Const ocColorGray2 As Long = &H50505
Public Const ocColorGray3 As Long = &H80808
Public Const ocColorGray4 As Long = &HA0A0A
Public Const ocColorGray5 As Long = &HD0D0D
Public Const ocColorGray6 As Long = &HF0F0F
Public Const ocColorGray7 As Long = &H121212
Public Const ocColorGray8 As Long = &H141414
Public Const ocColorGray9 As Long = &H171717
Public Const ocColorGray10 As Long = &H1A1A1A
Public Const ocColorGray11 As Long = &H1C1C1C
Public Const ocColorGray12 As Long = &H1F1F1F
Public Const ocColorGray13 As Long = &H212121
Public Const ocColorGray14 As Long = &H242424
Public Const ocColorGray15 As Long = &H262626
Public Const ocColorGray16 As Long = &H292929
Public Const ocColorGray17 As Long = &H2B2B2B
Public Const ocColorGray18 As Long = &H2E2E2E
Public Const ocColorGray19 As Long = &H303030
Public Const ocColorGray20 As Long = &H333333
Public Const ocColorGray21 As Long = &H363636
Public Const ocColorGray22 As Long = &H383838
Public Const ocColorGray23 As Long = &H3B3B3B
Public Const ocColorGray24 As Long = &H3D3D3D
Public Const ocColorGray25 As Long = &H404040
Public Const ocColorGray26 As Long = &H424242
Public Const ocColorGray27 As Long = &H454545
Public Const ocColorGray28 As Long = &H474747
Public Const ocColorGray29 As Long = &H4A4A4A
Public Const ocColorGray30 As Long = &H4D4D4D
Public Const ocColorGray31 As Long = &H4F4F4F
```

```
Public Const ocColorGray32 As Long = &H525252
Public Const ocColorGray33 As Long = &H545454
Public Const ocColorGray34 As Long = &H575757
Public Const ocColorGray35 As Long = &H595959
Public Const ocColorGray36 As Long = &H5C5C5C
Public Const ocColorGray37 As Long = &H5E5E5E
Public Const ocColorGray38 As Long = &H616161
Public Const ocColorGray39 As Long = &H636363
Public Const ocColorGray40 As Long = &H666666
Public Const ocColorGray41 As Long = &H696969
Public Const ocColorGray42 As Long = &H6B6B6B
Public Const ocColorGray43 As Long = &H6E6E6E
Public Const ocColorGray44 As Long = &H707070
Public Const ocColorGray45 As Long = &H737373
Public Const ocColorGray46 As Long = &H757575
Public Const ocColorGray47 As Long = &H787878
Public Const ocColorGray48 As Long = &H7A7A7A
Public Const ocColorGray49 As Long = &H7D7D7D
Public Const ocColorGray50 As Long = &H7F7F7F
Public Const ocColorGray51 As Long = &H828282
Public Const ocColorGray52 As Long = &H858585
Public Const ocColorGray53 As Long = &H878787
Public Const ocColorGray54 As Long = &H8A8A8A
Public Const ocColorGray55 As Long = &H8C8C8C
Public Const ocColorGray56 As Long = &H8F8F8F
Public Const ocColorGray57 As Long = &H919191
Public Const ocColorGray58 As Long = &H949494
Public Const ocColorGray59 As Long = &H969696
Public Const ocColorGray60 As Long = &H999999
Public Const ocColorGray61 As Long = &H9C9C9C
Public Const ocColorGray62 As Long = &H9E9E9E
Public Const ocColorGray63 As Long = &HA1A1A1
Public Const ocColorGray64 As Long = &HA3A3A3
Public Const ocColorGray65 As Long = &HA6A6A6
Public Const ocColorGray66 As Long = &HA8A8A8
Public Const ocColorGray67 As Long = &HABABAB
Public Const ocColorGray68 As Long = &HADADAD
Public Const ocColorGray69 As Long = &HB0B0B0
Public Const ocColorGray70 As Long = &HB3B3B3
Public Const ocColorGray71 As Long = &HB5B5B5
Public Const ocColorGray72 As Long = &HB8B8B8
Public Const ocColorGray73 As Long = &HBABABA
Public Const ocColorGray74 As Long = &HBDBDBD
Public Const ocColorGray75 As Long = &HBFBFBF
Public Const ocColorGray76 As Long = &HC2C2C2
Public Const ocColorGray77 As Long = &HC4C4C4
Public Const ocColorGray78 As Long = &HC7C7C7
Public Const ocColorGray79 As Long = &HC9C9C9
Public Const ocColorGray80 As Long = &HCCCCCC
Public Const ocColorGray81 As Long = &HCFCFCF
Public Const ocColorGray82 As Long = &HD1D1D1
Public Const ocColorGray83 As Long = &HD4D4D4
Public Const ocColorGray84 As Long = &HD6D6D6
Public Const ocColorGray85 As Long = &HD9D9D9
Public Const ocColorGray86 As Long = &HDBDBDB
Public Const ocColorGray87 As Long = &HDEDEDE
Public Const ocColorGray88 As Long = &HE0E0E0
Public Const ocColorGray89 As Long = &HE3E3E3
Public Const ocColorGray90 As Long = &HE5E5E5
Public Const ocColorGray91 As Long = &HE8E8E8
Public Const ocColorGray92 As Long = &HEBEBEB
```

```
Public Const ocColorGray93 As Long = &HEDEDED
Public Const ocColorGray94 As Long = &HF0F0F0
Public Const ocColorGray95 As Long = &HF2F2F2
Public Const ocColorGray96 As Long = &HF5F5F5
Public Const ocColorGray97 As Long = &HF7F7F7
Public Const ocColorGray98 As Long = &HFAFAFA
Public Const ocColorGray99 As Long = &HFCFCFC

'ocColorGreen As Long = &H00FF00
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorGreen As Long = 65280

Public Const ocColorGreenYellow As Long = &H2FFFAD
Public Const ocColorHoneydew As Long = &HF0FFF0
Public Const ocColorHotPink As Long = &HB469FF
Public Const ocColorIndianRed As Long = &H5C5CCD
Public Const ocColorIvory As Long = &HF0FFFF
Public Const ocColorKhaki As Long = &H8CE6F0
Public Const ocColorLavender As Long = &HFAE6E6
Public Const ocColorLavenderBlush As Long = &HF5F0FF

'ocColorLawnGreen As Long = &H00FC7C
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorLawnGreen As Long = 64636

Public Const ocColorLemonChiffon As Long = &HCDFAFF
Public Const ocColorLightBlue As Long = &HE6D8AD
Public Const ocColorLightCoral As Long = &H8080F0
Public Const ocColorLightCyan As Long = &HFFFFE0
Public Const ocColorLightGoldenrod As Long = &H82DDEE
Public Const ocColorLightGoldenrodYellow As Long = &HD2FAFA
Public Const ocColorLightGray As Long = &HD3D3D3
Public Const ocColorLightPink As Long = &HC1B6FF
Public Const ocColorLightSalmon As Long = &H7AA0FF
Public Const ocColorLightSeaGreen As Long = &HAAB220
Public Const ocColorLightSkyBlue As Long = &HFACE87
Public Const ocColorLightSlateBlue As Long = &HFF7084
Public Const ocColorLightSlateGray As Long = &H998877
Public Const ocColorLightSteelBlue As Long = &HDEC4B0
Public Const ocColorLightYellow As Long = &HE0FFFF
Public Const ocColorLimeGreen As Long = &H32CD32
Public Const ocColorLinen As Long = &HE6F0FA
Public Const ocColorMagenta As Long = &HFF00FF
Public Const ocColorMaroon As Long = &H6030B0
Public Const ocColorMediumAquamarine As Long = &HAACD66
Public Const ocColorMediumBlue As Long = &HCD0000
Public Const ocColorMediumOrchid As Long = &HD355BA
Public Const ocColorMediumPurple As Long = &HDB7093
Public Const ocColorMediumSeaGreen As Long = &H71B33C
Public Const ocColorMediumSlateBlue As Long = &HEE687B
Public Const ocColorMediumSpringGreen As Long = &H9AFA00
Public Const ocColorMediumTurquoise As Long = &HCCD148
Public Const ocColorMediumVioletRed As Long = &H8515C7
Public Const ocColorMidnightBlue As Long = &H701919
Public Const ocColorMintCream As Long = &HFAFFF5
Public Const ocColorMistyRose As Long = &HE1E4FF
Public Const ocColorMoccasin As Long = &HB5E4FF
Public Const ocColorNavajoWhite As Long = &HADDEFF
Public Const ocColorNavyBlue As Long = &H800000
Public Const ocColorOldLace As Long = &HE6F5FD
Public Const ocColorOliveDrab As Long = &H238E6B
```

```
'ocColorOrange As Long = &H00A5FF
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorOrange As Long = 42495

'ocColorOrangeRed As Long = &H0045FF
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorOrangeRed As Long = 17919

Public Const ocColorOrchid As Long = &HD670DA
Public Const ocColorPaleGoldenrod As Long = &HAAE8EE
Public Const ocColorPaleGreen As Long = &H98FB98
Public Const ocColorPaleTurquoise As Long = &HEEEEAF
Public Const ocColorPaleVioletRed As Long = &H9370DB
Public Const ocColorPapayaWhip As Long = &HD5EFFF
Public Const ocColorPeachPuff As Long = &HB9DAFF
Public Const ocColorPeru As Long = &H3F85CD
Public Const ocColorPink As Long = &HCBC0FF
Public Const ocColorPlum As Long = &HDDA0DD
Public Const ocColorPowderBlue As Long = &HE6E0B0
Public Const ocColorPurple As Long = &HF020A0
Public Const ocColorRed As Long = &HFF
Public Const ocColorRosyBrown As Long = &H8F8FBC
Public Const ocColorRoyalBlue As Long = &HE16941
Public Const ocColorSaddleBrown As Long = &H13458B
Public Const ocColorSalmon As Long = &H7280FA
Public Const ocColorSandyBrown As Long = &H60A4F4
Public Const ocColorSeaGreen As Long = &H578B2E
Public Const ocColorSeashell As Long = &HEEF5FF
Public Const ocColorSienna As Long = &H2D52A0
Public Const ocColorSkyBlue As Long = &HEBCE87
Public Const ocColorSlateBlue As Long = &HCD5A6A
Public Const ocColorSlateGray As Long = &H908070
Public Const ocColorSnow As Long = &HFAFAFF
Public Const ocColorSpringGreen As Long = &H7FFF00
Public Const ocColorSteelBlue As Long = &HB48246
Public Const ocColorTan As Long = &H8CB4D2
Public Const ocColorThistle As Long = &HD8BFD8
Public Const ocColorTomato As Long = &H4763FF
Public Const ocColorTurquoise As Long = &HD0E040
Public Const ocColorViolet As Long = &HEE82EE
Public Const ocColorVioletRed As Long = &H9020D0
Public Const ocColorWheat As Long = &HB3DEF5
Public Const ocColorWhite As Long = &HFFFFFF

'ocColorYellow As Long = &H00FFFF
'The above would be true, but Visual Basic removes the leading zeros
Public Const ocColorYellow As Long = 65535

Public Const ocColorYellowGreen As Long = &H32CD9A
```

```
Option Explicit
Dim fsa_cb  As Integer
Dim fsa_is As Integer
Dim fsa_bf As Integer
Dim fsa_tf As Integer
Dim fsa_ss As Integer

Dim PSW As Double
Dim PHW As Double


Dim Barray(0 To 2000) As Long
Dim NewData As Variant

'Storage for easier reference later
Dim NumRows As Long
Dim NumColumns As Long
Dim Sindex As Long

Dim X1 As Double
Dim X2 As Double
Dim x3 As Double
Dim x4 As Double
Dim x5 As Double
Dim x6 As Double
Dim x7 As Double
Dim x8 As Double
Dim x9 As Double

Dim SpineData As Double

Dim Stopit As Integer
Dim Head As Integer
Dim Feet As Integer
' class dim
Public FSASum As Double
Public FSAAverage As Double
Public FSASensors As Long
Public TorsoAverage As Double
Public TorsoSensors As Integer
Public ShoulderAverage As Double
Dim ShoulderWidth As Double
Public HipAverage As Double
Public WaistAverage As Double
Dim HipMaxWidth As Double
Dim WAverageWidth As Double
Public FSAWeight As Double
Dim FSAHeight As Double
Dim FSAIspring As Double
Dim TorsoLength As Double
Dim datacall As Long


Public Function Put_FSAData(ByVal element As Long, ByVal index As Long) As Double

    Dim i As Integer
    Dim x As Double
    On Error Resume Next
    Put_FSAData = -1

    If index < 0 Then Exit Function
```

```
    If index > 2000 Then Exit Function

    NewData = True

    x = 0

    Barray(index) = element
    If element Then datacall = datacall + 1

    For i = 0 To index
        x = x + Barray(i)
    Next i

    Put_FSAData = x

End Function


Public Property Get CBcoef() As Variant
    On Error Resume Next
    If NewData Then
        Call Crunchit
        NewData = False
    End If
    CBcoef = fsa_cb
End Property


Public Property Get IScoef() As Variant
    On Error Resume Next

    If NewData Then
        Call Crunchit
        NewData = False
    End If
    IScoef = fsa_is
End Property


Public Property Get BFcoef() As Variant
    On Error Resume Next

    If NewData Then
        Call Crunchit
        NewData = False
    End If
    BFcoef = fsa_bf
End Property


Public Property Get TFcoef() As Variant
    On Error Resume Next

    If NewData Then
        Call Crunchit
        NewData = False
    End If
    TFcoef = fsa_tf
End Property
```

```
Public Property Get SSprof() As Variant
    On Error Resume Next

    If NewData Then
        Call Crunchit
        NewData = False
    End If
    SSprof = fsa_ss
End Property

Private Sub Crunchit()

    Dim r As Integer
    Dim c As Integer
    Dim RAv As Double
    Dim avrav As Double
    Dim InARow As Integer
    Dim TorsoBottom As Integer
    Dim TorsoTop As Integer
    Dim TorsoLeft As Integer
    Dim TorsoRight As Integer
    Dim Brow As Integer
    Dim HipSum As Double
    Dim WaistSum As Double
    Dim HipSensors As Integer
    Dim WFirst As Integer
    Dim WLast As Integer
    Dim LastInARow As Integer
    Dim ShoulderSum As Double
    Dim ShoulderSensors As Integer
    Dim SLast As Integer
    Dim SFirst As Integer
    Dim Stretch As Integer
    Dim WaistSensors As Integer
    Dim DTemp As Double
    Dim TorsoCenter As Double
    Dim fsa As Integer
    Dim Center As Integer
    Dim Cfirst As Integer
    Dim darray(32, 32) As Double
    Dim delta As Double
    Dim Zeros As Integer
    Dim MCoef As Double
    Dim FCoef As Double
    Dim J As Double
    Dim U1(1 To 9) As Double
    Dim U2(1 To 9) As Double
    Dim Z1(1 To 9) As Double
    Dim Z1_2(1 To 9) As Double
    Dim Z2(1 To 9) As Double
    Dim Z2_2(1 To 9) As Double
    Dim Z1_Z2(1 To 9) As Double
    Dim Y1(1 To 9) As Double

    Dim SumU2 As Double
    Dim SumZ1_2 As Double
    Dim SumZ2_2 As Double
    Dim SumZ1_Z2 As Double
    Dim SumY1_Z1 As Double
    Dim SumY1_Z2 As Double
    Dim divisor As Double
```

```
    Dim S As Double
    Dim Lumbar As Double

    Dim CData(1 To 32, 1 To 32)
    Dim UnitMultiplier As Double

    On Error Resume Next

    UnitMultiplier = 0.392156862745098

    fsa = 0
    FSASum = 0
    FSASensors = 0

    NumColumns = 32
    NumRows = 32

    For c = 1 To NumColumns
        For r = 1 To NumRows

            DTemp = Barray(fsa) * UnitMultiplier
            CData(c, r) = DTemp
            If DTemp < 50 Then
                If DTemp Then
                    FSASum = FSASum + DTemp
                    FSASensors = FSASensors + 1
                End If
            End If
            fsa = fsa + 1
        Next r

    Next c

    If FSASensors = 0 Then GoTo error_out

    'standard stat stuff
    FSAAverage = FSASum / FSASensors

    FSAWeight = FSASum * 0.03

    ' if the person is less than 80 lbs - abort
    If FSAWeight < 80 Then GoTo error_out

    'let's have a swipe at some more stats
    ' ie height, hips, waist, & shoudlers


    'set up first cb factor..

    fsa_cb = FSAWeight * 3.5
'    If FSAWeight < 200 Then fsa_cb = 2
'    If FSAWeight < 150 Then fsa_cb = 1

    Stopit = False
    For c = 1 To NumColumns
        For r = 1 To NumRows
            If CData(c, r) Then Stopit = True
            If Stopit Then Exit For
        Next r
        If Stopit Then Exit For
    Next c
```

```
    Head = c

    Stopit = False
    For c = NumColumns To 1 Step -1
        For r = NumRows To 1 Step -1
            If CData(c, r) Then Stopit = True
            If Stopit Then Exit For
        Next r
        If Stopit Then Exit For
    Next c

    Feet = c

    FSAHeight = (2 + (Feet - Head)) * 2

    If FSAAverage = 0 Then GoTo error_out

    FSAIspring = FSAWeight / FSAAverage

    fsa_is = FSAIspring * 100

'    If FSAIspring < 8.5 Then fsa_is = 2
'    If FSAIspring < 6.5 Then fsa_is = 1

    'Find the "TorsoBottom" for use in calculations
    Stopit = False
    For c = NumColumns - 5 To 1 Step -1
        For r = 1 To NumRows
            If CData(c, r) Then
                InARow = InARow + 1
                If InARow > 12 Then Stopit = True
             Else:
                InARow = 0
            End If
            If Stopit Then
                Exit For
            End If
        Next r
        If Stopit Then Exit For
    Next c

    TorsoBottom = c + 1

    Center = 0
    Cfirst = 0
    Stopit = False
    For r = 1 To NumRows
        If CData(c, r) Then
            Center = Center + 1
            If Cfirst = 0 Then Cfirst = r
        End If
    Next r


    TorsoCenter = Cfirst + Center / 2

    'Find the "TorsoTop" for use in caluculations

    Stopit = 0
    For c = TorsoBottom - 10 To 1 Step -1
```

```
        If c > 32 Then GoTo error_out
        If c < 1 Then GoTo error_out
        For r = 1 To NumRows
            If CData(c, r) Then
                InARow = InARow + 1
                If InARow < Stopit Then Stopit = InARow
            Else: InARow = 0
            End If
        Next r
        If Stopit < 5 Then Exit For
        Stopit = 0
    Next c

    If c < 1 Then c = 1

    TorsoTop = c

    'Now that we have located TorsoBottom and TorsoTop
    'Find the shoulder width by slicing the torso data lengthwise

    'From the Right:

    For r = 1 To NumRows / 2
        For c = TorsoTop To TorsoTop + 3
            If c > 32 Then GoTo error_out
            If c < 1 Then GoTo error_out
            If CData(c, r) Then
                InARow = InARow + 1
                If InARow > 1 Then Exit For
            Else:
                InARow = 0
            End If
        Next c
        If InARow > 1 Then Exit For
    Next r
    TorsoRight = r

    'From the Left:

    For r = NumRows To NumRows / 2 Step -1
        For c = TorsoTop To TorsoTop + 3
            If c > 32 Then GoTo error_out
            If c < 1 Then GoTo error_out
            If CData(c, r) Then
                InARow = InARow + 1
                If InARow > 1 Then Exit For
                Else: InARow = 0
                End If
        Next c
        If InARow > 1 Then Exit For
    Next r

    TorsoLeft = r


    ShoulderWidth = ((TorsoLeft - TorsoRight) * 0.75) + 3

    Brow = TorsoBottom - 4
```

```
    'Hip & Waist average

    LastInARow = 0
    InARow = 0
    WFirst = 0
    WLast = 0

    HipSum = 0
    WaistSum = 0
    HipSensors = 0
    HipMaxWidth = 0
    HipAverage = 0
    WaistAverage = 0
    WAverageWidth = 0
    WaistSensors = 0

'    Dim ct As Integer

    For c = Brow To Brow + 4
        For r = Cfirst To NumRows

            If CData(c, r) Then
                HipSum = HipSum + CData(c, r)
                HipSensors = HipSensors + 1
                InARow = InARow + 1
                If InARow > LastInARow Then
                    LastInARow = InARow
                End If
            Else: InARow = 0
            End If


            If c - 6 < 1 Then GoTo error_out
            If CData(c - 6, r) Then

                WaistSum = WaistSum + CData(c - 5, r)
                WaistSensors = WaistSensors + 1

            End If

            If c - 5 < 1 Then GoTo error_out
            If CData(c - 5, r) Then
                If WFirst = 0 Then WFirst = r
                WLast = r
            End If
        Next r
        WAverageWidth = WAverageWidth + (WLast - WFirst) * 0.75
        WFirst = 0
    Next c


    If HipSensors = 0 Then HipSensors = HipSensors + 1

    If WaistSensors = 0 Then WaistSensors = WaistSensors + 1

    HipMaxWidth = LastInARow * 0.75
    HipAverage = HipSum / HipSensors
    WaistAverage = WaistSum / WaistSensors
    WAverageWidth = WAverageWidth / 4
```

```
    'Shoulder average

    ShoulderSum = 0
    ShoulderSensors = 0
    ShoulderAverage = 0
    SFirst = 32
    SLast = 0
    Stretch = 0
    InARow = 0

    For c = TorsoTop To TorsoTop + 3
        If c > 32 Then GoTo error_out
        If c < 1 Then GoTo error_out

        For r = TorsoRight To TorsoLeft
            If r > 32 Then GoTo error_out
            If r < 1 Then GoTo error_out
            If CData(c, r) Then

                ShoulderSum = ShoulderSum + CData(c, r)
                ShoulderSensors = ShoulderSensors + 1

            End If
        Next r
    Next c

    If ShoulderSensors = 0 Then GoTo error_out

    ShoulderAverage = ShoulderSum / ShoulderSensors


    TorsoAverage = 0
    TorsoSensors = 0

    'Calculate the average for the entire torso

    For c = TorsoTop To TorsoBottom
        If c > 32 Then GoTo error_out
        If c < 1 Then GoTo error_out
        For r = TorsoRight To TorsoLeft
            If r > 32 Then GoTo error_out
            If r < 1 Then GoTo error_out
            If CData(c, r) < 80 Then
                If CData(c, r) Then
                    TorsoAverage = TorsoAverage + CData(c, r)
                    TorsoSensors = TorsoSensors + 1
                End If
            End If
        Next r
    Next c

    If TorsoSensors = 0 Then GoTo error_out

    TorsoLength = (TorsoBottom - TorsoTop) * 2

    TorsoAverage = TorsoAverage / TorsoSensors

    'Predicted (Shoulder Width)/Weight = 0.14538 - 0.00000613*(Total mmHg) +
    '0.0007852*(Average) - 0.0005343*(SWidth) - 0.0007978*(TLength)

    'Predicted (Hip Width)/Weight = 0.12607 +13.358*(HWidth/Total mmHg) -
```

```
    '0.0009497*(SWidth) - 0.0020362*(HWidth) - 0.0015309*(TLength)


'    PSW = 0.14538 - 0.00000613 * FSASum + 0.0007852 * FSAAverage - 0.0005343 * ShoulderWidth -
0.0007978 * TorsoLength
'    PHW = 0.12607 + 13.358 * (HipMaxWidth / FSASum) - 0.0009497 * ShoulderWidth - 0.0020362 * H
ipMaxWidth - 0.0015309 * TorsoLength

'    PSW = 1 / (PSW + PHW)



    fsa_bf = (((FSAWeight / ShoulderWidth) + (FSAWeight / HipMaxWidth)) / 2) * 45


'    fsa_bf = (ShoulderAverage + HipAverage) * 10

'    If PSW < 7.5 Then fsa_bf = 3
'    If PSW < 6.5 Then fsa_bf = 2
'    If PSW < 5.5 Then fsa_bf = 1


    Lumbar = WaistAverage

    fsa_tf = Lumbar * 10

'    If Lumbar < 35 Then fsa_tf = 2
'    If Lumbar < 25 Then fsa_tf = 1

    Lumbar = WaistAverage

    fsa_ss = 0
    If Lumbar > 15 Then fsa_ss = 1
    If Lumbar > 20 Then fsa_ss = 2
    If Lumbar > 23 Then fsa_ss = 3
    If Lumbar > 28 Then fsa_ss = 4
    If Lumbar > 32 Then fsa_ss = 5
    If Lumbar > 38 Then fsa_ss = 6
    If Lumbar > 42 Then fsa_ss = 7
    If Lumbar > 45 Then fsa_ss = 8
    If Lumbar > 50 Then fsa_ss = 9


    GoTo end_sub

error_out:
    fsa_cb = 0
    fsa_is = 0
    fsa_bf = 0
    fsa_tf = 0
    fsa_ss = 5
end_sub:

    Exit Sub


 End Sub
```

```
'*************************************************************************
'*
'* Copyright (c) 1998, KL GROUP INC.  All Rights Reserved.
'* http://www.klg.com
'*
'* This file is provided for demonstration and educational uses only.
'* Permission to use, copy, modify and distribute this file for
'* any purpose and without fee is hereby granted, provided that the
'* above copyright notice and this permission notice appear in all
'* copies, and that the name of KL Group not be used in advertising
'* or publicity pertaining to this material without the specific,
'* prior written permission of an authorized representative of
'* KL Group.
'*
'* KL GROUP MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY
'* OF THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
'* TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
'* PURPOSE, OR NON-INFRINGEMENT. KL GROUP SHALL NOT BE LIABLE FOR ANY
'* DAMAGES SUFFERED BY USERS AS A RESULT OF USING, MODIFYING OR
'* DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
'*
'*************************************************************************
Option Explicit

Public Type TRACKER
    Original As Boolean
    Linear As Boolean
End Type

Public Type POINTAPI
    x As Long
    y As Long
End Type

#If Win32 Then
    Public Declare Function CreatePen Lib "gdi32" (ByVal nPenStyle As Long, ByVal nWidth As Long
, ByVal crColor As Long) As Long
    Public Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long
    Public Declare Function GetDC Lib "user32" (ByVal hWnd As Long) As Long
    Public Declare Function ReleaseDC Lib "user32" (ByVal hWnd As Long, ByVal hDc As Long) As Lo
ng
    Public Declare Function Rectangle Lib "gdi32" (ByVal hDc As Long, ByVal X1 As Long, ByVal Y1
 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
    Public Declare Function SelectObject Lib "gdi32" (ByVal hDc As Long, ByVal hObject As Long)
As Long
    Public Declare Function SetROP2 Lib "gdi32" (ByVal hDc As Long, ByVal nDrawMode As Long) As
Long
#Else
    Public Declare Function CreatePen Lib "gdi" (ByVal nPenStyle As Integer, ByVal nWidth As Int
eger, ByVal crColor As Long) As Integer
    Public Declare Function DeleteObject Lib "gdi" (ByVal hObject As Integer) As Integer
    Public Declare Function GetDC Lib "user" (ByVal hWnd As Integer) As Integer
    Public Declare Function Rectangle Lib "gdi" (ByVal hDc As Integer, ByVal X1 As Integer, ByVa
l Y1 As Integer, ByVal X2 As Integer, ByVal Y2 As Integer) As Integer
    Public Declare Function ReleaseDC Lib "user" (ByVal hWnd As Integer, ByVal hDc As Integer) A
s Integer
    Public Declare Function SelectObject Lib "gdi" (ByVal hDc As Integer, ByVal hObject As Integ
er) As Integer
    Public Declare Function SetROP2 Lib "gdi" (ByVal hDc As Integer, ByVal nDrawMode As Integer)
 As Integer
#End If
```

```
Private FSAType As String
Private FSAReserve As Long
Private CalFile1 As String
Private MapFlag As Integer
Private NumArrays As Integer
Private Width As Integer
Private Height As Integer
Public UnitMultiplier As Double
Private Units As String
Private Label As String
Private SizeOfFrame As Long
Public NumberofFrames As Integer
Public Junk As String
Public FSAData(0 To 2000) As Byte



Sub ReadFile(filename)

    Open filename For Binary Access Read As #1
    FSAType = String(6, " ")
    CalFile1 = String(48, " ")

    Get #1, 1, FSAType
    Get #1, , FSAReserve
    Get #1, , CalFile1
    Get #1, , MapFlag
    If MapFlag Then

    End If

    Get #1, , NumArrays
    Get #1, , Width
    Get #1, , Height
    Get #1, , UnitMultiplier
    Units = String(6, " ")
    Label = String(32, " ")
    Get #1, , Units
    Get #1, , Label

    If NumArrays = 2 Then

    End If
    Get #1, , SizeOfFrame
    Get #1, , NumberofFrames
    Junk = String(10, " ")
    Get #1, , Junk
    FSASum = 0
    FSAWeight = 0
    FSASensors = 0
    For i = 0 To SizeOfFrame - 9
        Get #1, , FSAData(i)
    Next i

    Close #1
End Sub

'Sub NextFrame(FileName)

'    Open FileName For Binary Access Read As #1
```

VistaFile - 2

```
'    For i = NextFrame To NextFrame + SizeOfFrame - 9
'        Get #1, i, FSAData(i)
'    Next i

'    Close #1
'End Sub
```

Calibrate - 1

```
Option Explicit

Private Sub CancelButton_Click()
    'call ApiAbort
    Unload Me
End Sub

Private Sub Form_Load()

    Call ApiCalibrate

End Sub

Private Sub OKButton_Click()
    Unload Me
End Sub

Private Sub Timer1_Timer()

    Dim Complete As Variant

    Status.Caption = "Calibrating"

    If ProgressBar1.Value = 100 Then

        ProgressBar1.Value = 0
        Timer1.Enabled = False
        Call GetCals
        Complete = ApiComplete(0, "CALIBRATE")
        If Complete = &HAABFF Then
            Status.Caption = "Complete"
            Exit Sub
        End If
        Timer1.Enabled = True

    End If

    ProgressBar1.Value = ProgressBar1.Value + 1

End Sub

Private Sub GetCals()

    Dim i As Integer

    For i = 0 To 19

    ' need a good way to filter out unused axes
    ' for now - hard code it for back lying

        If i <> 12 And i <> 14 And i <> 16 And i <> 18 And i <> 10 Then

            Text1(i).Text = Axis.ValueOf(i + 1, "CAL")

        End If
    Next i

End Sub
```

EntryError - 1

```
Option Explicit

Private Sub OKButton_Click()
    Unload Me
End Sub
```

```
EStop - 1

Private Done As Integer

Private Sub Form_Load()

    Done = False

    Call ApiEstop

    Done = True

End Sub

Private Sub Timer1_Timer()

   Dim Complete As Long

   If ProgressBar1.Value = 100 Then

        ProgressBar1.Value = 0
        Timer1.Enabled = False

        If Done = True Then
            Status.Caption = "Test Complete"
            Unload Me
        End If

        Timer1.Enabled = True

   End If

   ProgressBar1.Value = ProgressBar1.Value + 1
End Sub
```

```
FileError - 1

Private Sub Accept_Click()
    Main.Choice = 0
    Unload Me
End Sub

Private Sub Cancel_Click()
    Main.Choice = 1
    Unload Me
End Sub
```